Αλγόριθμοι και Πολυπλοκότητα – εξετάσεις εαρινού εξαμήνου 2021-2022

Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Πανεπιστημίου Ιωαννίνων

27/6/2022

Διάρκεια εξέτασης: 2 ώρες

Θέμα Α [1,1,1]

1. Σχεδιάστε πρόχειρα σε ένα σύστημα αξόνων τις συναρτήσεις: n, loglogn, logn, n^2. Γραμμοσκιάστε την περιοχή στην οποία αντιστοιχεί το Ω(n).

2. Τι πολυπλοκότητα έχει ο ακόλουθος κώδικας; Εξηγήστε γιατί.

```
def fun(data, value):
    n = len(data)
    left = 0
    right = n - 1
    while left <= right:
        middle = (left + right) // 2
        if value < data[middle]:
            right = middle - 1
        elif value > data[middle]:
            left = middle + 1
        else:
            return middle
    raise ValueError('Value is not in the list')

data1 = [...] # μια λίστα με n τιμές
data2 = [...] # μια λίστα με m τιμές
results = []
for value in data1:
    result.append(fun(data2, value))
```

3. Γράψτε αποδοτικό κώδικα (σε Python ή σε ψευδοκώδικα) που να υπολογίζει το πλήθος εμφανίσεων των λατινικών γραμμάτων σε ένα δεδομένο κείμενο στα αγγλικά.

Θέμα Β [1,1,1]

1. Περιγράψτε το πρόβλημα της μέγιστης υποακολουθίας (max subarray) και έναν τρόπο επίλυσης του προβλήματος με πολυπλοκότητα O(n^2) ή καλύτερη. Δώστε ένα αριθμητικό παράδειγμα που να επιδεικνύει τον τρόπο επίλυσης που προτείνετε.

2. Δίνεται η ακόλουθη φράση «SUMMER HOLIDAYS». Χρησιμοποιήστε την κωδικοποίηση Huffman για την κωδικοποίηση της (σε ισοβαθμίες δώστε προτεραιότητα στο γράμμα που προηγείται αλφαβητικά).

3. Δίνεται μια λίστα εργασιών με γνωστούς χρόνους έναρξης και τερματισμού. Κάθε εργασία μπορεί να εκτελεστεί σε μια μηχανή και υπάρχουν πολλές διαθέσιμες ίδιες μηχανές. Γράψτε κώδικα (ή ψευδοκώδικα) που να αναθέτει βέλτιστα τις εργασίες σε μηχανές έτσι ώστε να χρησιμοποιούνται συνολικά οι λιγότερες δυνατές μηχανές.

Θέμα Γ [1,0.5,0.5]

1. Δίνεται η ακόλουθη λίστα τιμών: 4,1,5,6,7,2,9,8. Δείξτε σχηματικά την ταξινόμηση με quicksort που χρησιμοποιεί ως pivot το πλέον δεξιό στοιχείο. Τι πολυπλοκότητα χρόνου έχει ο αλγόριθμος quick sort;

2. Συμπληρώστε τον ακόλουθο κώδικα (αντικαταστήστε το pass) έτσι ώστε αναδρομικά να υπολογίζει το γινόμενο μόνο των περιττών στοιχείων μιας λίστας.

```
def prodr(a):
    pass

a = […] # μια λίστα με ακέραιες τιμές
print(prodr(a))
```

3. Με βάση το MASTER θεώρημα

$T(n) = \begin{cases} c & \text{if } n < d \\ aT(n/b) + f(n) & \text{if } n \geq d \end{cases}$	1. if $f(n)$ is $O(n^{\log_b a - \varepsilon})$, then $T(n)$ is $\Theta(n^{\log_b a})$ 2. if $f(n)$ is $\Theta(n^{\log_b a} \log^k n)$, then $T(n)$ is $\Theta(n^{\log_b a} \log^{k+1} n)$ 3. if $f(n)$ is $\Omega(n^{\log_b a + \varepsilon})$, then $T(n)$ is $\Theta(f(n))$, provided $af(n/b) \leq \delta f(n)$ for some $\delta < 1$.

υπολογίστε την πολυπλοκότητα των ακόλουθων αναδρομικών εξισώσεων.

I. T(n) = 16T(n/2) + n^2

II. T(n) = T(n/2) + 1

Θέμα Δ [0.5,0.5,1]

1. Περιγράψτε λεκτικά το πως μπορεί να χρησιμοποιηθεί ο αλγόριθμος DFS για τον εντοπισμό ενός κύκλου σε έναν κατευθυνόμενο γράφο.

2. Τι σημαίνει το χαρακτηριστικό «βελτιστότητα υποπροβλημάτων» στο δυναμικό προγραμματισμό; Αναφέρατε 2 προβλήματα που λύνονται αποδοτικά με δυναμικό προγραμματισμό.

3. Τι πρόβλημα επιλύουν ο αλγόριθμος του Prim και ο αλγόριθμος του Kruskal; Εφαρμόστε και τους δύο αλγόριθμους στο ακόλουθο γράφημα βήμα προς βήμα καταγράφοντας τη σειρά με την οποία προσαρτούνται οι ακμές στο αποτέλεσμα