Αλγόριθμοι και Πολυπλοκότητα – εξετάσεις χειμερινού εξαμήνου 2021-2022

Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Πανεπιστημίου Ιωαννίνων

1/2/2022

Διάρκεια εξέτασης: 2 ώρες

Θέμα Α [1,1,0.5]

1. Σχεδιάστε πρόχειρα σε ένα σύστημα αξόνων τις 4 συναρτήσεις: n, n^2, logn, 2^n. Γραμμοσκιάστε την περιοχή στην οποία αντιστοιχεί το O(logn).

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

```
def fun(n):
    c = 0
    while n>0:
        for _ in range(n):
            c += 1
        n = n // 2
```

3. Σε μια επεκτάσιμη ακολουθία που αντιγράφεται όταν γεμίζει σε μια νέα ακολουθία με διπλάσιο μέγεθος, ποια είναι η πολυπλοκότητα χρόνου χειρότερης περίπτωσης για την πράξη της εισαγωγής και ποια η επιμερισμένη πολυπλοκότητα χρόνου για την ίδια πράξη.

Θέμα Β [0.5,1,1]

1. Τι θα εμφανίσει ο ακόλουθος κώδικας σε Python;

```
import heapq

li = [6, 2, 5, 1, 7]
heapq.heapify(li)
while len(li) > 0:
    print(heapq.heappop(li))
```

Παρατήρηση: heapq είναι σωρός ελαχίστων στην Python

2. Δίνεται μια ακολουθία A με 1.000.000 τιμές και μια ακολουθία B με 100 τιμές. Γράψτε αποδοτικό κώδικα (σε Python ή σε ψευδοκώδικα) που θα εξετάζει αν οι ακολουθίες είναι ξένες μεταξύ τους. Ποια είναι η πολυπλοκότητα του κώδικά σας;

3. Ποιες λειτουργίες υποστηρίζει ο αφηρημένος τύπος ξένων συνόλων (disjoint sets); Δώστε ένα παράδειγμα προβλήματος στο οποίο η συγκεκριμένη δομή μπορεί να δώσει υπολογιστικό πλεονέκτημα.

Θέμα Γ [1,1,1]

1. Δίνεται η ακόλουθη λίστα τιμών: 4,1,5,6,7,2,9,8. Δείξτε σχηματικά α) την ταξινόμηση με merge sort και β) την ταξινόμηση με quicksort που χρησιμοποιεί ως pivot το πλέον αριστερό στοιχείο. Τι πολυπλοκότητα χρόνου χειρότερης περίπτωσης έχει ο καθένας από τους αλγόριθμους merge sort και quick sort;

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

```
def sumr(a):
    pass

a = [1,2,3,4,5]
print(sumr(a))
```

3. Με βάση το MASTER θεώρημα

$T(n)=\begin{cases} c & \text{if } n<d \\ aT(n/b)+f(n) & \text{if } n\ge d \end{cases}$	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 $af(n/b)\le \delta f(n)$ for some $\delta<1$.

υπολογίστε την πολυπλοκότητα των ακόλουθων αναδρομικών εξισώσεων.

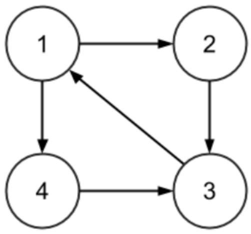
I. T(n) = 2T(n/2) + 1
II. T(n) = 2T(n/2) + n^2

Θέμα Δ [1,0.5,0.5]

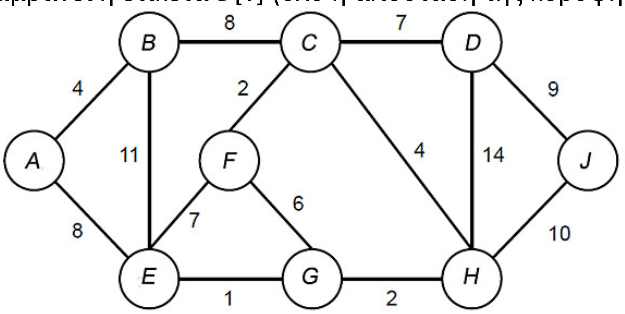
1. Δίνονται οι δύο συμβολοσειρές X=HORSEBACK και Y=SNOWFLAKE. Συμπληρώστε τον ακόλουθο πίνακα με τον οποίο ο αλγόριθμος δυναμικού προγραμματισμού LCS εντοπίζει τη μέγιστη κοινή ακολουθία ανάμεσα στις δύο συμβολοσειρές.

			H	O	R	S	E	B	A	C	K
		-1	0	1	2	3	4	5	6	7	8
	-1										
S	0										
N	1										
O	2										
W	3										
F	4										
L	5										
A	6										
K	7										
E	8										

2. Δείξτε την αναπαράσταση του ακόλουθου γραφήματος α) με πίνακα γειτονικότητας και β) με λίστα γειτονικότητας.



3. Εφαρμόστε τον αλγόριθμο του Dijkstra με εκκίνηση την κορυφή A για την εύρεση των συντομότερων διαδρομών στον ακόλουθο γράφο. Καταγράψτε στον πίνακα που ακολουθεί για κάθε κορυφή v όλες τις τιμές που σταδιακά λαμβάνει η ετικέτα D[v] (δλδ η απόσταση της κορυφής v από την κορυφή αφετηρίας).



Κορυφή v	D[v]
A	0
B	4
C	
D	
E	8
F	
G	
H	
J	