Αλγόριθμοι και Πολυπλοκότητα

Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Πανεπιστημίου Ιωαννίνων

1/9/2022, διάρκεια εξέτασης 3 ώρες

Θέμα Α [0.5, 0.5, 0.3, 0.2]

1. Εξηγήστε εάν και πότε ένας αλγόριθμος πολυπλοκότητας O(n^2) μπορεί να εκτελείται ταχύτερα από έναν αλγόριθμο πολυπλοκότητας O(nlogn) στο ίδιο υπολογιστικό σύστημα.
2. Ποιες από τις ακόλουθες 5 συναρτήσεις είναι O(logn);
n, n^2, nlogn, 5logn, logn + 1000
3. Τι χρονική πολυπλοκότητα έχει ένας αλγόριθμος εύρεσης της μεγαλύτερης τιμής σε έναν πίνακα που πρώτα ταξινομεί σε φθίνουσα διάταξη τον πίνακα με τον αλγόριθμο bubble sort (ταξινόμηση φυσαλίδας) και στη συνέχεια επιστρέφει την πρώτη τιμή του πίνακα;
4. Ποιο είναι το άθροισμα της σειράς 2+3+4+5+…+n;

Θέμα Β [0.5, 0.5, 1, 1]

1. Τι πολυπλοκότητα έχει ο ακόλουθος αλγόριθμος που έχει κωδικοποιηθεί στην python ως συνάρτηση με όνομα fun; Εξηγήστε γιατί.

```
def fun(myList):
    if len(myList) > 1:
        mid = len(myList) // 2
        left = myList[:mid]
        right = myList[mid:]
        fun(left)
        fun(right)
        i,j,k = 0, 0, 0
        while i < len(left) and j < len(right):
            if left[i] <= right[j]:
                myList[k] = left[i]
                i += 1
            else:
                myList[k] = right[j]
                j += 1
            k += 1
        while i < len(left):
            myList[k] = left[i]
            i += 1
            k += 1
        while j < len(right):
            myList[k]=right[j]
            j += 1
            k += 1
```

2. Σε μια επεκτάσιμη ακολουθία που αντιγράφεται όταν γεμίζει, σε μια νέα ακολουθία με μέγεθος κατά 1 μεγαλύτερο, ποια είναι η **επιμερισμένη πολυπλοκότητα χρόνου** για την πράξη της εισαγωγής και γιατί; Ποια καλύτερη λύση θα προτείνατε;
3. Χρησιμοποιώντας ένα σωρό ελαχίστων, συμπληρώστε τον ακόλουθο κώδικα έτσι ώστε να ταξινομεί σε φθίνουσα σειρά τα στοιχεία μιας λίστα, υλοποιώντας τη διαδικασία της ταξινόμησης σωρού (heapsort). Παρατήρηση: Η συνάρτηση **`heapq.heapify(li)`** μετατρέπει τη λίστα **`li`** σε σωρό ελαχίστων και η συνάρτηση **`heapq.heappop(li)`** επιστρέφει το κορυφαίο στοιχείο του σωρού.

```
import heapq
li = [10,11,22,16,19,23,12]

heapq.heapify(li)
```

4. Δίνεται μια λίστα με τηλέφωνα συνδρομητών που κάλεσαν μια υπηρεσία. Γράψτε αποδοτικό κώδικα (σε Python ή σε ψευδοκώδικα) που να εντοπίζει το τηλέφωνο ή τα τηλέφωνα από τα οποία έγιναν οι περισσότερες κλήσεις. Τι πολυπλοκότητα έχει ο αλγόριθμος που προτείνετε και γιατί;

Θέμα Γ [1, 0.5, 0.5, 1]

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 δεξί στοιχείο. Τι πολυπλοκότητα χρόνου χειρότερης περίπτωσης και τι πολυπλοκότητα μέσης περίπτωσης έχει ο αλγόριθμος;

2. Συμπληρώστε τον ακόλουθο κώδικα (αντικαταστήστε το pass) έτσι ώστε αναδρομικά να υπολογίζει το ελάχιστο στοιχείο μιας λίστας.

```
def minr(a):
    pass
a = [15, 2, 7, 1, 16, 5]
print(minr(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.

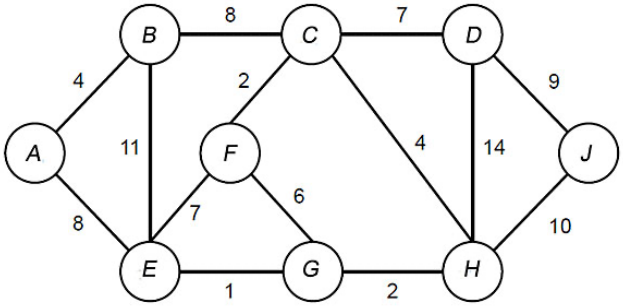
I. T(n) = T(n/4) + logn
II. T(n) = 2T(n/2) + n

4. Ποιες είναι οι λειτουργίες που υποστηρίζει ο αφηρημένος τύπος δεδομένων (ΑΤΔ) **ξένων συνόλων** και με ποιο άλλο όνομα είναι γνωστός; Χρησιμοποιώντας τον ΑΤΔ **ξένων συνόλων** επιλύστε το ακόλουθο πρόβλημα και περιγράψτε λεκτικά τη λύση που προτείνετε. Δίνεται ένα σύνολο από σχέσεις φιλίας ανάμεσα σε άτομα (κάθε άτομο αναπαρίσταται από έναν αριθμό και η σχέση φιλίας με ένα ζεύγος αριθμών). Εντοπίστε τις ομάδες ατόμων που είναι συνδεδεμένες (π.χ. friends = [[1, 0], [2, 3], [1, 2], [4, 5]])

Θέμα Δ [1, 1, 0.5]

1. Έστω ένα σακίδιο με ικανότητα μεταφοράς W μονάδων βάρους και τα ακόλουθα αντικείμενα. Αντικείμενο 1 με βάρος w1 μονάδων και αξία v1, αντικείμενο 2 με βάρος w2 μονάδων και αξία v2, αντικείμενο 3 με βάρος w3 και αξία v3. Προτείνετε τρόπο υπολογισμού της βέλτιστης λύσης του προβλήματος σε περίπτωση που κάθε αντικείμενο μπορεί είτε να επιλεγεί ολόκληρο είτε όχι. Ποια αλγοριθμική τεχνική χρησιμοποιείται για τη γενική περίπτωση ν αντικειμένων; Θα αλλάξει και πως ή λύση αν μπορούν να επιλεγούν αυθαίρετα μικρά τμήματα από κάθε αντικείμενο; Ποια αλγοριθμική τεχνική χρησιμοποιείται τότε για τη γενική περίπτωση ν αντικειμένων;

2. Για τον ακόλουθο γράφο εντοπίστε το ελάχιστο συνεκτικό δένδρο χρησιμοποιώντας τον αλγόριθμο του Prim. Θεωρώντας ότι ο αλγόριθμος ξεκινά από την κορυφή Α, καταγράψτε τις ακμές που ενσωματώνονται στο ελάχιστο συνεκτικό δένδρο με τη σειρά που συμβαίνει αυτό. Επίσης, σχεδιάστε και το τελικό συνεκτικό δένδρο.



3. Εφαρμόστε τον αλγόριθμο του Dijkstra για την εύρεση της συντομότερης διαδρομής από την κορυφή Α στην κορυφή J για το παραπάνω γράφημα. Απαντήστε **καταγράφοντας μόνο τη σειρά με την οποία εξετάζονται οι κορυφές και τοποθετούνται στη λίστα των «λυμένων» κορυφών** (δηλαδή έχει προσδιοριστεί για αυτές η συντομότερη διαδρομή από την αφετηρία)