Αλγόριθμοι και Πολυπλοκότητα – Ιούνιος 2025

Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Πανεπιστημίου Ιωαννίνων

4/6/2025 (διάρκεια εξέτασης 2,5 ώρες)

Θέμα 1 [A=1, B=1]

Α. Κάνοντας καταγραφή των πρωτογενών λειτουργιών υπολογίστε τη συνάρτηση χρόνου εκτέλεσης T(n) για τον ακόλουθο αλγόριθμο (εύρεση του ελαχίστου στοιχείου πίνακα), όπου n είναι το μέγεθος της εισόδου. Αναφέρατε ποιες λειτουργίες θεωρήσατε ως πρωτογενείς.

```
Algorithm FindMin(A, n)
min <- A[1]
For i <- 2 to n
   If A[i] < min
      min <- A[i]
return min
```

Β. 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 της αναδρομικής συνάρτησης **T(n) = 4T(n/4) + n**.

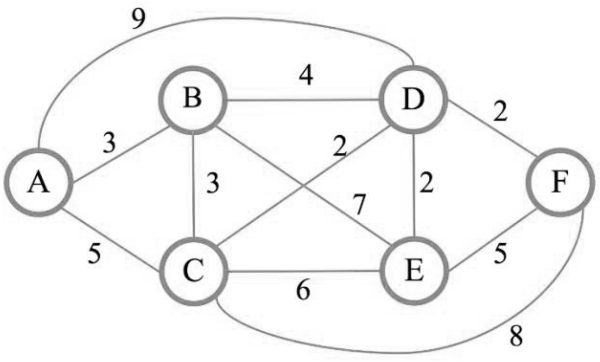
Θέμα 2 [A=2, B=2]

Α. Δίνεται ένας πίνακας A που περιέχει n ακέραιους.

1. Περιγράψτε το πρόβλημα του μέγιστου υποπίνακα (Max Subarray Sum) για τον πίνακα A.
2. Περιγράψτε τον αλγόριθμο του Kadane για την επίλυση του προβλήματος.
3. Ποια είναι η πολυπλοκότητά του και γιατί θεωρείται βέλτιστος;
4. Δοθέντος του πίνακα A = [3, -2, 5, -1, 6, -3, 2, -5, 4] εκτελέστε τον αλγόριθμο του Kadane συμπληρώνοντας τον ακόλουθο πίνακα στη σειρά που αφορά τα επιθεματικά αθροίσματα:

Ακολουθία τιμών (A)		**3**	**-2**	**5**	**-1**	**6**	**-3**	**2**	**-5**	**4**
Επιθεματικά αθροίσματα (M)	**0**									
Δείκτες (t)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

Β. Υπολογίστε το ελάχιστο συνεκτικό δέντρο για τον διπλανό γράφο α) με τον αλγόριθμο του Prim ξεκινώντας από την κορυφή A και β) με τον αλγόριθμο του Kruskal. Και για τους δύο αλγόριθμους καταγράψτε τις ακμές που ορίζουν το ελάχιστο συνεκτικό δέντρο με τη σειρά με την οποία ο αλγόριθμος τις προσαρτά σε αυτό.



Θέμα 3 [A=2, B=2]

Α. Δίνεται η συνάρτηση κατακερματισμού **h(k) = k mod 7** και ένας πίνακας κατακερματισμού ανοικτής διευθυνσιοδότησης μεγέθους 7.

1. Εισάγετε στον πίνακα κατακερματισμού τα κλειδιά 10, 20, 15, 7, 22, 35 με τη σειρά που δίνονται, χρησιμοποιώντας γραμμική ανίχνευση συγκρούσεων (linear probing). Δείξτε την τελική μορφή (μόνο) που θα έχει ο πίνακας κατακερματισμού μετά την εισαγωγή όλων των κλειδιών.

0	1	2	3	4	5	6

2. Αν μετά την εισαγωγή των κλειδιών, αναζητηθεί το κλειδί 17, ποιες θέσεις του πίνακα κατακερματισμού θα έχουν ελεγχθεί μέχρι να διαπιστωθεί ότι το κλειδί δεν υπάρχει στον πίνακα κατακερματισμού;

Β. Συμπληρώστε τα 8 κενά στον παρακάτω κώδικα για την υλοποίηση πίνακα κατακερματισμού ανοικτής διευθυνσιοδότησης. Θεωρείστε ότι η συνάρτηση κατακερματισμού είναι η **h(k) = k mod 7** και κάθε στοιχείο είναι ένας ακέραιος που αποθηκεύεται στον πίνακα κατακερματισμού ως έχει (δηλαδή key=value). Θεωρείστε ότι αν ένα στοιχείο διαγράφεται τότε εισάγεται στη θέση που βρισκόταν στον πίνακα κατακερματισμού η τιμή DELETED.

```
hash_table = [None] * 7

def hash_func(key):
    return ___1____

def insert(key):
    index = hash_func(key)
    for i in range(len(hash_table)):
        probe_index = (index + i) % len(hash_table)
        if hash_table[probe_index] is None or hash_table[probe_index] == ___2____:
            hash_table[probe_index] = key
            return
        elif hash_table[probe_index] == ___3____:
            return  # Το στοιχείο υπάρχει ήδη

def contains(key):
    index = hash_func(key)
    for i in range(len(hash_table)):
        probe_index = (index + i) % len(hash_table)
        if hash_table[probe_index] is None:
            return ___4____
        elif hash_table[probe_index] == key:
            return ___5____
    return ___6____

def remove(key):
    index = ___7____
    for i in range(len(hash_table)):
        probe_index = (index + i) % len(hash_table)
        if hash_table[probe_index] is None:
            return
        elif hash_table[probe_index] == key:
            hash_table[probe_index] = ___8____
            return

insert(10)
insert(20)
insert(15)
print(contains(20))   # Αναμένεται: True
remove(10)
print(contains(10))   # Αναμένεται: False
```