# HY240: Δομές Δεδομένων
# Εαρινό Εξάμηνο – Ακαδημαϊκό Έτος 2017-18
# Παναγιώτα Φατούρου

## 4ο Σετ Ασκήσεων

**Ημερομηνία Παράδοσης:** Τρίτη, 29 Μαΐου 2018

**Τρόπος Παράδοσης:** Οι ασκήσεις μπορούν να παραδοθούν σε ηλεκτρονική μορφή χρησιμοποιώντας το πρόγραμμα turnin (δείτε οδηγίες στην ιστοσελίδα του μαθήματος), είτε στους βοηθούς του μαθήματος, την Δευτέρα, 28 Μαΐου 2018, ώρα 13:00-14:00. Ασκήσεις που παραδίδονται μετά τις 14:00 της Δευτέρας, 28/05/2018 θα θεωρούνται εκπρόθεσμες. Εκπρόθεσμες ασκήσεις γίνονται δεκτές σε ηλεκτρονική μορφή και η αποστολή τους πρέπει να γίνει χρησιμοποιώντας το πρόγραμμα turnin (το αρχείο που θα αποστείλετε πρέπει να είναι σε μορφή pdf). **Ασκήσεις μέσω ηλεκτρονικού ταχυδρομείου δεν γίνονται δεκτές.**

## Ασκήσεις

1. Δίνονται τα ακόλουθα στοιχεία: 8, 15, 22, 17, 29, 31, 10. Να τα εισάγετε με τη σειρά που δίνονται σε έναν πίνακα κατακερματισμού 7 θέσεων, δεδομένου ότι ως μέθοδος διαχείρισης συγκρούσεων χρησιμοποιείται: [15%]
   α. η μέθοδος των μικτών αλυσίδων με συνάρτηση κατακερματισμού $h_1(x) = x \bmod 7$ και κελάρι τριών θέσεων που δεικτοδοτείται από τους αριθμούς -1, -2, και -3 (δηλαδή ο πίνακας έχει συνολικά 10 θέσεις σ΄ αυτήν την περίπτωση).
   β. διπλός κατακερματισμός με πρωτεύουσα συνάρτηση κατακερματισμού $h_1(x) = x \bmod 7$ και δευτερεύουσα συνάρτηση κατακερματισμού $h_2(x) = x^2 \bmod 7$ (πίνακας 7 θέσεων),
   γ. ταξινομημένος κατακερματισμός ανοικτής διεύθυνσης. Θεωρήστε ότι οι συναρτήσεις κατακερματισμού είναι ίδιες με αυτές του ερωτήματος ii. (πίνακας 7 θέσεων)

2. Θεωρήστε ένα πίνακα κατακερματισμού που αποθηκεύει ακέραια κλειδιά χωρίς πρόσημο. Τα κλειδιά είναι των 32 bit και είναι πάντα δυνάμεις του 2. Βρείτε ποιο είναι το ελάχιστο μέγεθος t του πίνακα κατακερματισμού και παρουσιάστε συνάρτηση κατακερματισμού που παίρνει ως όρισμα ένα κλειδί *x* και παράγει ένα νούμερο μεταξύ 1 και *t*, έτσι ώστε να μην συμβαίνουν ποτέ συγκρούσεις. [15%]

3. Έστω T ένα δυαδικό δένδρο. Περιγράψτε αλγόριθμο για να το μετατρέψετε σε AVL. Ο αλγόριθμός σας πρέπει να έχει χρονική πολυπλοκότητα O(nlogn). Τροποποιείστε τον αλγόριθμό σας (αν χρειάζεται) ώστε να εκτελείται σε χρόνο O(n) στην καλύτερη περίπτωση. [20%]

4. Θεωρήστε ένα σωρό H και τον ακόλουθο αλγόριθμο:

```
BST-From-Min-Heap(H)
1   T = New-Empty-BinarySearchTree()
2   for i = 1 to length of H
3         BST-Tree-Insert(T,H[i])
4   return T
```

Αποδείξτε (με τη μέθοδο του παραδείγματος) ότι ο αλγόριθμος BST-From-Min-Heap δεν παράγει πάντα δένδρο δυαδικής αναζήτησης (binary search tree) ελάχιστου ύψους. [15%]

5. Να παρουσιαστεί αλγόριθμος που να δέχεται ως είσοδο έναν δείκτη στη ρίζα ενός δένδρου AVL και να τυπώνει το μακρύτερο μονοπάτι του δένδρου. Ο αλγόριθμός σας θα πρέπει να έχει χρονική πολυπλοκότητα O(logn), όπου n είναι το πλήθος των κόμβων του δένδρου. [15%]

6. Σχεδιάστε εκ νέου τη λειτουργία HeapInsert() ενός σωρού ώστε αυτή να διενεργείται από πάνω προς τα κάτω. Συγκεκριμένα, η HeapInsert() θα πρέπει να κάνει κατάλληλα swap κλειδιών καθώς διασχίζει κόμβους από τη ρίζα προς τον κόμβο που θα αποτελέσει τον πατρικό κόμβο του νεοεισαχθέντα κόμβου. Οι ενέργειες αυτές θα πρέπει να πραγματοποιούνται έτσι ώστε, μετά την εισαγωγή, ο νεοεισαχθέντας κόμβος να αποτελέσει το δεξιότερο φύλλο και να μην χρειαστούν περαιτέρω ενέργειες (π.χ. swap) μετά την τοποθέτηση του νέου κόμβου στον πίνακα που υλοποιεί το σωρό. Η έκδοση της HeapInsert() που θα σχεδιάσετε θα πρέπει να έχει χρονική πολυπλοκότητα O(logn) και να διατηρεί τις ιδιότητες του σωρού. [20%]