ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΡΧΕΙΩΝ

2η άσκηση

Ημερομηνία παράδοσης: 3 Μαΐου 2022

Δυαδικό Δένδρο Έρευνας (5 μονάδες)

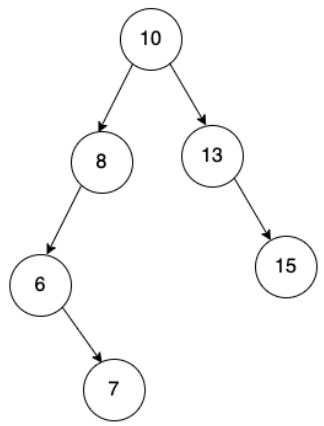
Κατασκευάστε την κλάση «Δυαδικό Δένδρο Έρευνας» (ΔΔΕ) με (α) χρήση πεδίου αριθμών (array) μεγέθους N και (β) με χρήση δυναμικής παραχώρησης μνήμης. Εκτός από τις βασικές πράξεις info(tree_pointer), left(tree_pointer), right(tree_pointer) που ισχύουν γενικά για ΔΔΕ, η κλάση υποστηρίζει τις πράξεις:

- Εισαγωγή τυχαίου κλειδιού
- Αναζήτηση τυχαίου κλειδιού
- Διαγραφή τυχαίου κλειδιού

Για την υλοποίηση (α) θα χρησιμοποιήσετε ένα πεδίο Nx3 ακεραίων αριθμών (N είναι ο αριθμός των στοιχείων). Κάθε θέση του πίνακα παριστάνει έναν κόμβο του δένδρου και σε κάθε μία από τις 3 στήλες αποθηκεύονται τα πεδία info, left, right του κόμβου[1].

	info	left	right
0			7
1	13	NULL	6
2			0
3	7	NULL	NULL
4	10	5	1
5	8	8	NULL
6	15	NULL	NULL
7			9
8	6	NULL	3
9			NULL

ROOT = 4



AVAIL = 2

Available Positions
2
0
7
9
NULL
NULL
NULL
NULL
NULL
NULL

Το παραπάνω σχήμα (αριστερά), δείχνει της υλοποίηση του ΔΔΕ (κέντρο). Η ρίζα του δένδρου βρίσκεται στην θέση 4 (ROOT= 4). Η υλοποίηση είναι αντίστοιχη με την υλοποίηση συνδεδεμένης

[1] Ένα πεδίο μεγέθους N στοιχείων μπορεί να παραστήσει ένα ΔΔΕ με N στοιχεία το πολύ.

λίστας με array[2]. Το πεδίο left έχει τιμή που δηλώνει την θέση του αριστερού υποδένδρου. Το πεδίο right έχει διπλό ρόλο: (α) δηλώνει την θέση του δεξιού υποδένδρου και (β) χρησιμοποιείται επίσης για να υλοποιήσει την στοίβα (stack) με τις διαθέσιμες θέσεις του array. Στο σχήμα (δεξιά), η θέση AVAIL=2 δηλώνει ότι θέση 2 είναι η επόμενη διαθέσιμη θέση που θα χρησιμοποιηθεί στην επόμενη εισαγωγή κόμβου. Αν γίνει διαγραφή κόμβου, τότε αυτή θα προστεθεί στην κορυφή της στοίβας (και η θέση AVAIL θα πάρει τιμή την θέση του κόμβου που διαγράφτηκε). Επομένως, για να υλοποιήσετε τις μεθόδους εισαγωγής και διαγραφής στοιχείων από τον ΔΔΕ θα χρειαστείτε πιο πριν να έχετε υλοποιήσει της μεθόδους getnode(tree_pointer) και free_node(tree_pointer) στην στοίβα των ελεύθερων θέσεων του array[3]. Στο παραπάνω σχήμα (δεξιά) φαίνεται η μορφή της στοίβας που υλοποιεί η στήλη right του array. Προσέξετε ότι δεν είναι ανάγκη να φτιάξετε την στοίβα με ξεχωριστή δομή δεδομένων έξω από το array (η στοίβα στα δεξιά του σχήματος υπάρχει ήδη στο πεδίο right).

Χρησιμοποιήστε ένα πεδίο με Nx3 και κάντε εισαγωγή N κλειδιών. Σας δίνονται δυο αρχεία κλειδιών με 100 και 10^6 κλειδιά (θα σας δοθούν τις επόμενες μέρες). Τα 10^6 κλειδιά θα τα χρησιμοποιήσετε για να φτιάξετε το δένδρο. Τα 100 κλειδιά επιλέγονται μέσα από τα 10^6 και θα τα χρησιμοποιήσετε για αναζητήσεις και διαγραφές. Τις παρακάτω μετρήσεις θα τις επαναλάβετε και για τις δύο υλοποιήσεις.

1. Μετρήστε το μέσο αριθμό συγκρίσεων ανά εισαγωγή στην διάρκεια κατασκευής του ΔΔΕ.
2. Κάντε 100 διαγραφές των 100 αριθμών που σας δίνονται και μετρήστε τον μέσο αριθμό συγκρίσεων ανά διαγραφή. Η πράξη διαγραφής καλεί πρώτα την πράξη αναζήτησης.
3. Τον συνολικό χρόνο για N εισαγωγές.
4. Τον συνολικό χρόνο για 100 διαγραφές.

Συμπληρώστε τις τιμές στο παρακάτω πίνακα. Δικαιολογήστε την απόδοση κάθε μεθόδου. Δώστε ιδιαίτερη βαρύτητα στην ερμηνεία των αποτελεσμάτων και μην δείξετε μόνο τις τιμές.

Μέθοδος	**Μέσος αριθμός συγκρίσεων / εισαγωγή (10^6 εισαγωγές)**	**Συνολικός χρόνος για 10^6 εισαγωγές**	**Μέσος αριθμός συγκρίσεων / ανά διαγραφή (100 διαγραφές)**	**Συνολικός χρόνος για 100 διαγραφές**
ΔΔΕ με δυναμική παραχώρηση μνήμης				
ΔΔΕ με array				

<u>Συστάσεις:</u>

- Την υλοποίηση (β) θα την βρείτε στο Web και βιβλία και θα την χρησιμοποιήσετε ως έχει. Θα χρειαστεί να επέμβετε στην υλοποίηση ώστε να υπολογίζει τον αριθμό συγκρίσεων ανά πράξη εισαγωγής ή αναζήτησης και για να μετρήσετε τον χρόνο[4].
- Μην προχωρήσετε στην υλοποίηση πριν καταλάβετε καλά την υλοποίηση συνδεδεμένης λίστας με array που συζητήθηκε στο μάθημα.
- Δώστε προσοχή στην αρχικοποίηση της λίστας διαθέσιμων θέσεων. Θα πρέπει όλες οι θέσεις του πεδίου να βρίσκονται στην στοίβα των διαθέσιμων θέσεων. Η αρχικοποίηση

[2] Συμβουλευτείτε την υλοποίηση της σελ. 16 της διάλεξης για συνδεδεμένες λίστες.
[3] Η υλοποίηση της στοίβας γίνεται όπως και στην περίπτωση της συνδεδεμένης λίστας.
[4] https://www.tutorialspoint.com/how-to-measure-elapsed-time-in-nanoseconds-with-java

μπορεί να γίνει με τον ίδιο ακριβώς τρόπο όπως στην συνδεδεμένη λίστα. Με τον ίδιο τρόπο επίσης υλοποιούνται οι μέθοδοι getnode και freenode (αντί για next χρησιμοποιείστε τον δείκτη right).

- Το πρόγραμμά σας πρέπει να δουλέψει για την είσοδο κλειδιών σε αρχείο που θα δώσουμε εμείς κατά την διόρθωση της άσκησης.
- Η άσκηση πρέπει να τυπώνει στην οθόνη τις τιμές του παραπάνω πίνακα για οποιοδήποτε αρχείο κλειδιών που θα δίνεται στην είσοδο.

Ουρά προτεραιότητας (5 μονάδες)

Κατασκευάστε μια ουρά προτεραιότητας (heap) με (α) με χρήση πεδίου αριθμών και (β) με δυναμική παραχώρηση μνήμης. Υλοποιήστε τις πράξεις

- εισαγωγή τυχαίου κλειδιού
- διαγραφή (μέγιστου) κλειδιού
- κατασκευή ουράς προτεραιότητας όταν τα κλειδιά δίνονται το ένα μετά το άλλο.
- κατασκευή ουράς προτεραιότητας όταν τα κλειδιά δίνονται όλα μαζί.

Χρησιμοποιήστε τα ίδια αρχεία κλειδιών του προηγούμενου ερωτήματος. Κατασκευάστε την heap με τις μεθόδους (α) και (β) και μετρήστε τον συνολικό χρόνο κατασκευής και με τις δύο μεθόδους. Στην συνέχεια κάντε διαγραφή 100 κλειδιών και μετρήστε τον μέσο αριθμό συγκρίσεων ανά διαγραφή. Συμπληρώστε τον παρακάτω πίνακα μετρήσεων.

Δώστε ιδιαίτερη βαρύτητα στην ερμηνεία των αποτελεσμάτων και μην δείξετε μόνο τις τιμές. Δικαιολογήστε την απόδοση κάθε μεθόδου. Δώστε ιδιαίτερη βαρύτητα στην ερμηνεία των αποτελεσμάτων και μην δείξετε μόνο τις τιμές.

Μέθοδος	**Συνολικός χρόνος κατασκευής όταν τα κλειδιά δίνονται όλα μαζί**	**Συνολικός χρόνος κατασκευής όταν τα κλειδιά εισάγονται το ένα μετά το άλλο**	**Μέσος αριθμός συγκρίσεων / εισαγωγή (10^6 εισαγωγές)**	**Μέσος αριθμός συγκρίσεων / ανά διαγραφή (100 διαγραφές)**	**Συνολικός χρόνος για 100 διαγραφές**
Ουρά προτεραιότητας με array					
Ουρά προτεραιότητας με Δυναμική Παραχώρησης	**(**)**				

Συστάσεις:

- Την υλοποίηση (α) θα τη βρείτε στις διαλέξεις του μαθήματος για δένδρα, στο Web ή σε βιβλία και θα την χρησιμοποιήσετε ως έχει.
- Την υλοποίηση (β) θα την κάνετε εσείς. Δεν θα την βρείτε στο Web.

- Μην προχωρήσετε στην υλοποίηση πριν καταλάβετε καλά την υλοποίηση ουράς προτεραιότητας με array που συζητήθηκε στο μάθημα.
- Ισχύει πάντα ότι η ουρά προτεραιότητας αντιστοιχεί σε πλήρες δυαδικό δένδρο δηλαδή υπάρχουν όλοι οι κόμβοι, σε όλα τα επίπεδα εκτός των φύλλων όπου μπορεί να λείπουν οι κόμβοι δεξιά
- Για την υλοποίηση με δυναμική παραχώρηση μνήμης θα χρειαστείτε (α) δείκτη προς τον πατέρα κάθε κόμβου, (β) δείκτη στο τελευταίο στοιχείο της ουράς προτεραιότητας (γ) για τον έλεγχο ορθότητας θα χρειαστείτε μια συνάρτηση που θα τυπώνει το δυαδικό δένδρο και μπορείτε να την βρείτε στο Web[5].
- Για την κατασκευή της ουράς προτεραιότητας με δυναμική παραχώρηση μνήμης στην περίπτωση που δίνονται όλα τα στοιχεία, θα εισάγετε τα στοιχεία σε ένα δυαδικό δένδρο με τυχαία σειρά. Για την μετατροπή του δένδρου σε ουρά προτεραιότητας, συστήνεται να χρησιμοποιήσετε μία λίστα που κρατά τους δείκτες στα υποδένδρα. Αν οι δείκτες εισάγονται στην αρχή της λίστας, τότε θα έχετε πρόσβαση στα υποδένδρα με την σειρά που πρέπει να τα επεξεργαστείτε.
- (**) Για το συγκεκριμένο ερώτημα θα ακολουθήσει διευκρινιστική ανακοίνωση.

[5] https://www.geeksforgeeks.org/print-binary-tree-2-dimensions/

Παραδοτέα: Ένα συμπιεσμένο zip αρχείο που περιέχει ότι ζητείται παρακάτω:

- Ο κώδικας περιέχει συνοπτικά σχόλια που εξηγούν την υλοποίηση.
- Μία έκθεση που περιγράφει σε 1-2 σελίδες πως φτιάχτηκε ο κώδικας (δηλ. για κάθε ερώτημα ποια είναι η γενική ιδέα της λύσης σε 3-4 προτάσεις), υπάρχουν σαφείς οδηγίες μετάφρασης από compiler και εκτέλεσης, τι λάθη έχει (αν έχει, περιπτώσεις που δεν δουλεύει το πρόγραμμα, ή περιπτώσεις που κάνει περισσότερα από όσα σας ζητεί η άσκηση, τι χρησιμοποιήσατε από έτοιμα προγράμματα ή πηγές πληροφόρησης. Υποδείξετε ακόμα και πηγές στο WWW όπως Wikipedia (πλήρεις διευθύνσεις) ή ακόμα και συναδέλφους που σας βοήθησαν στην άσκηση.
- Προσπαθήστε να δικαιολογήστε την απόδοση κάθε μεθόδου. Εξηγήστε για ποιο λόγο μία μέθοδος είναι πιο αποδοτική από μία άλλη για αναζήτηση. Συνολικά δεν χρειάζεται να γράψετε πάνω από 2 σελίδες αρκεί κάθε απάντηση να είναι σαφής και αιτιολογημένη σωστά. Δεν χρειάζεται να γράψετε περισσότερα σχόλια για την υλοποίηση του κώδικα εκτός από ότι ζητείται παρακάτω (δείτε την ενότητα «παραδοτέα»).
- Στην ενότητα τεκμηρίωσης των αποτελεσμάτων πρέπει να υπάρχει απόλυτη σαφήνεια
- Εκτός των παραπάνω, οι ασκήσεις βαθμολογούνται με άριστα εφόσον:
    - Το zip είναι πλήρες
    - Οι κώδικες περνούν από compiler και εκτελούνται κανονικά και σωστά σε windows ή Linux περιβάλλον
    - Ο κώδικάς σας δουλεύει για οποιοδήποτε αρχείο δεδομένων που θα σας δοθεί ως είσοδος, και όχι μόνο το αρχείο που παράγετε εσείς

keys_1000000_BE.bin
4.000.000 bytes που αντιστοιχούν σε 1.000.000 μοναδικούς ακεραίους με πρόσημο, Big Endian κωδικοποίηση, 4 byte ανά ακέραιο. Ο πρώτος αριθμός είναι το 1, οπότε αν διαβάσετε τον πρώτο αριθμό, και δεν είναι 1, κάνετε κάτι λάθος.
Θα χρησιμοποιήσετε το αρχείο για τα κλειδιά που θα εισάγετε στις δομές.

keys_100_BE.bin
Περιέχει 100 κλειδιά τα οποία υπάρχουν στο αρχείο των κλειδιών που θα εισάγετε. 2 από αυτά τα κλειδιά είναι ίδια. Οπότε εάν στους 1.000.000 κόμβους κάνετε αυτές τις 100 διαγραφές, το δέντρο/heap σας θα έχει 999.001 κλειδιά.