ΔΕΥΤΕΡΗ ΕΡΓΑΣΙΑ

Στην εργασία αυτή θα υλοποιήσετε ένα δυαδικό δέντρο αναζήτησης στο οποίο εφαρμόζονται τεχνικές επαναζύγισης αν χρειάζεται μετά από κάθε εισαγωγή διαγραφή στο δέντρο. Συγκεκριμένα, το δέντρο αυτό έχει δύο παραμέτρους εισόδου c και b ($c > 1$ και $b > 0$) και μετά από κάθε εισαγωγή ή διαγραφή, θα εξασφαλίζεται ότι το δέντρο έχει ύψος το πολύ $\lceil c\log_2(n+1) + b \rceil$ όπου n είναι το τρέχον πλήθος των στοιχείων μετά τη λειτουργία της εισαγωγής ή διαγραφής αντίστοιχα. Υπενθυμίζεται επίσης ότι το βάθος $\mathrm{depth(v)}$ ενός κόμβου v είναι το πλήθος των κόμβων από τη ρίζα στο v ενώ το ύψος του v, $\mathrm{height(v)}$ είναι το πλήθος των κόμβων στο μακρύτερο μονοπάτι από αυτόν τον κόμβο σε φύλλο του δέντρου (που βρίσκεται στο υποδέντρο του v). Το ύψος ενός δέντρου είναι το ύψος της ρίζας του δέντρου.

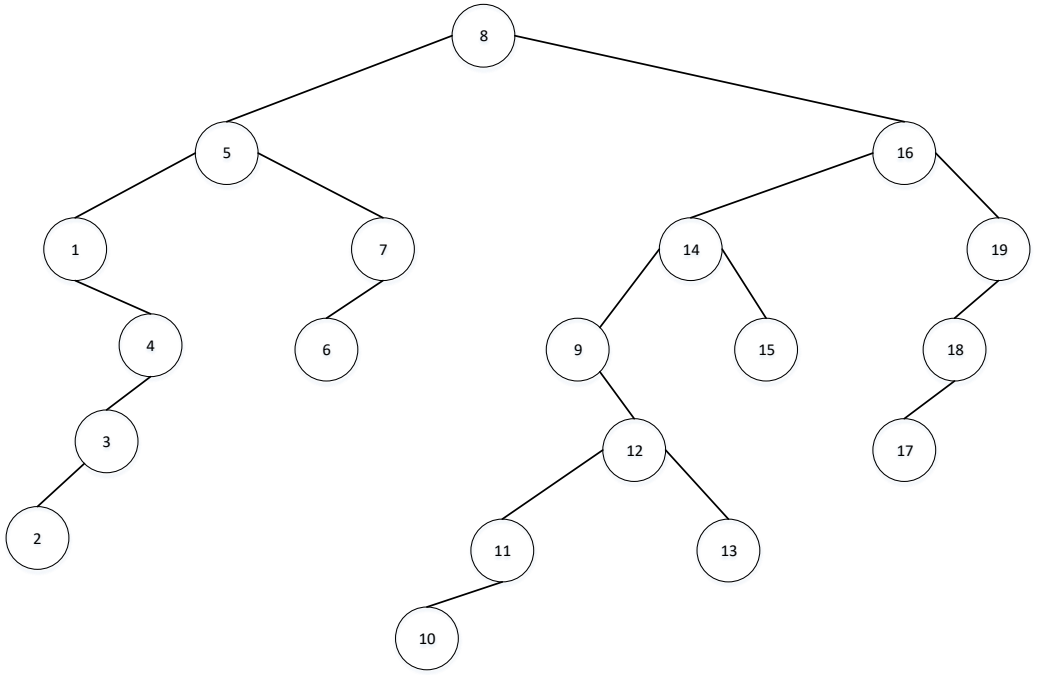
Οι λειτουργίες της αναζήτησης, εισαγωγής και διαγραφής ενός στοιχείου, υλοποιούνται όπως και σε ένα κλασικό δυαδικό δέντρο αναζήτησης. Όπως θα αποσαφηνιστεί παρακάτω, το δέντρο ανά τακτά χρονικά διαστήματα, ανακατασκευάζεται πλήρως . Έστω d είναι το πλήθος των διαγραφών που έχουν γίνει μέχρι μία δεδομένη χρονική στιγμή από την τελευταία φορά που το δέντρο ανακατασκευάστηκε πλήρως.

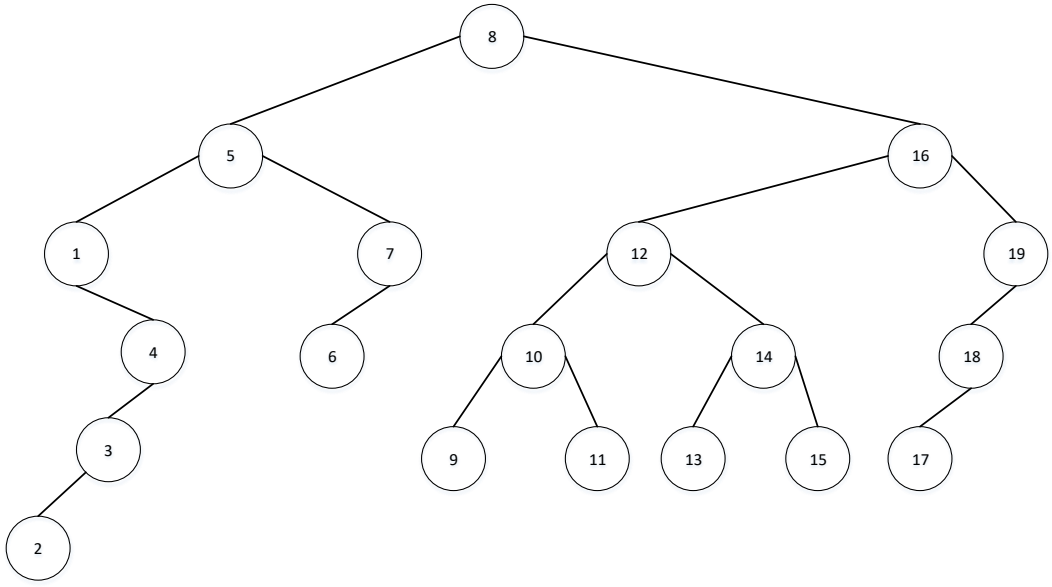
Πιο αναλυτικά, μετά από κάθε λειτουργία ενημέρωσης, γίνονται οι εξής ενέργειες.

- *Εισαγωγή*: Αν w είναι το νέο φύλλο και $\mathrm{depth(w)} > \lceil c\log_2(n+1+\mathrm{d}) \rceil$, ανεβαίνουμε το μονοπάτι προς τη ρίζα μέχρι να βρούμε τον πρώτο κόμβο v τέτοιον ώστε $\mathrm{height(v)} > \lceil c\log_2(n_v+1) \rceil$ όπου n_v είναι το πλήθος των κόμβων του υποδέντρου με ρίζα το v. Στη συνέχεια, γίνεται πλήρης ανακατασκευή αυτού του υποδέντρου ώστε το νέο υποδέντρο να είναι δυαδικό δέντρο αναζήτησης ελαχίστου ύψους για τα n_v στοιχεία που περιέχονται σε αυτό το δέντρο. Για την υλοποίηση αυτής της τεχνικής επαναζύγισης δεν θα πρέπει να χρησιμοποιήσετε επιπλέον πεδία στους κόμβους του δέντρου παρά μόνο αυτά που απαιτούνται για την υλοποίηση ενός κλασικού δυαδικού δέντρου αναζήτησης. Για την ανάβαση στο δέντρο, μπορείτε να χρησιμοποιήσετε μία βοηθητική δομή μεγέθους $\mathrm{depth(w)}$ στοιχείων για να αποθηκεύσετε το μονοπάτι από τη ρίζα στο φύλλο w. Όσον αφορά την ανακατασκευή του υποδέντρου, μπορείτε να χρησιμοποιήσετε ένα βοηθητικό πίνακα n_v στοιχείων αν το κρίνετε απαραίτητο, αν και υπάρχουν και τεχνικές που υλοποιούν την ανακατασκευή αυτή χωρίς τη χρήση βοηθητικού πίνακα, κατευθείαν στο αρχικό δέντρο (με τη χρήση πολλαπλών περιστροφών). Συνολικά, ο χρόνος όλης αυτής της επεξεργασίας μετά την εισαγωγή ενός στοιχείου θα πρέπει να είναι $\Theta(n_v)$.
- *Διαγραφή*: Αυξάνουμε το μετρητή d κατά ένα. Αν $d \geq (2^{\frac{b}{c}} - 1)(n+1)$ τότε ανακατασκευάζεται από την αρχή όλο το δέντρο έτσι ώστε να προκύψει δυαδικό δέντρο αναζήτησης ελαχίστου ύψους για τα n στοιχεία. Σε αυτή την περίπτωση ο μετρητής d τίθεται ίσος με μηδέν. Επίσης το κόστος της ανακατασκευής πρέπει να είναι $\Theta(\mathrm{n})$ ενώ και πάλι μπορείτε να χρησιμοποιήσετε ένα βοηθητικό πίνακα n στοιχείων για αυτή την επεξεργασία.

Στα σχήματα που ακολουθούν, φαίνονται οι αλλαγές που συμβαίνουν μετά από τη εισαγωγή του 10 σε ένα δέντρο αυτού του είδους. Θεωρούμε ότι το $c = 1{,}2$ *και το* $\mathrm{d} = 5$ δηλ. έχουν γίνει ήδη πέντε διαγραφές από την τελευταία πλήρη ανακατασκευή του δέντρου. Με την εισαγωγή του 10 το βάθος

του 10, $\mathrm{depth}(10)$, είναι ίσο με 7 που είναι μεγαλύτερο της ποσότητας $\lceil cl\log_2(n+1+\mathrm{d})\rceil = \lceil 1{,}2l\log_2(19+1+5)\rceil = 6$. Στη συνέχεια, ανεβαίνουμε το μονοπάτι προς τη ρίζα από το φύλλο 10 μέχρι να φτάσουμε στον κόμβο 14 που είναι πρώτος κόμβος για τον οποίο ισχύει $he\mathrm{ight}(14) = 5 > \lceil 1{,}2l\log_2(7+1)\rceil$. Το επόμενο βήμα είναι να ανακατασκευάσουμε πλήρως το υποδέντρο με ρίζα τον κόμβο 14 και το αποτέλεσμα φαίνεται στο δεύτερο κατά σειρά σχήμα.





Το σημαντικό πλεονέκτημα που προσφέρει αυτή η δομή δεδομένων είναι ότι παρόλο που κάποιες λειτουργίες έχουν υψηλό κόστος, εντούτοις υπό το πρίσμα μίας μακράς ακολουθίας λειτουργιών ενημέρωσης, το μέσο κόστος είναι πολύ χαμηλότερο. Πιο αναλυτικά, έστω $I_1, I_2, \ldots, I_m$ είναι m λειτουργίες ενημέρωσης όπου η λειτουργία I_{i} μπορεί να είναι μία λειτουργία εισαγωγής ή διαγραφής.

Αν $c(I_i)$ είναι το κόστος λειτουργίας I_i το οποίο συμπεριλαμβάνει και το κόστος των επανορθωτικών ενεργειών που τυχόν απαιτούνται μετά τη λειτουργία I_i , τότε για το συνολικό κόστος των m λειτουργιών ισχύει:

$$\sum_{\mathrm{i=1}}^{m} c(I_i) = \mathrm{O}(\sum_{i=1}^{m} \log_2(n_i))$$

όπου n_i είναι το πλήθος των στοιχείων του δέντρου μετά την ολοκλήρωση της λειτουργίας I_i. Ουσιαστικά, η παραπάνω σχέση δείχνει ότι αν και κάποιες λειτουργίες μπορεί να έχουν πολύ υψηλό ή χαμηλό κόστος, λαμβάνοντας υπόψη το συνολικό κόστος των m λειτουργιών, μπορούμε να θεωρήσουμε ότι κάθε λειτουργία I_i εκτελείται σε χρόνο $O(\log_2(n_i))$ δηλ. έχει χρονική πολυπλοκότητα ίδιας τάξης με αυτή ενός ισοζυγισμένου δυαδικού δέντρου αναζήτησης που περιέχει n_i στοιχεία. Το κόστος $O(\log_2(n_i))$ είναι γνωστό ως επιμερισμένο κόστος (amortized cost) της λειτουργίας I_i.

Συνοψίζοντας, καλείστε να υλοποιήσετε τις τρεις βασικές λειτουργίες της αναζήτησης, εισαγωγής και διαγραφής στο συγκεκριμένο αυτό δέντρο.

ΠΑΡΑΔΟΤΕΑ

Θα πρέπει να παραδοθεί ο πηγαίος κώδικας εκτυπωμένος καθώς και σε CD μαζί με τον εκτελέσιμο κώδικα. Ιδιαίτερη βαρύτητα θα πρέπει να δοθεί στη σωστή τεκμηρίωση του προγράμματός σας. Θα πρέπει λοιπόν ο κώδικας σας να συνοδεύεται από ξεχωριστό κείμενο που θα παρέχει λεπτομερή περιγραφή των τεχνικών σας. Επίσης, εντός του πηγαίου κώδικα θα πρέπει να υπάρχουν «πυκνά» σχόλια διατυπωμένα στα **ελληνικά**.

Η εργασία μπορεί να εκπονηθεί από ομάδα μέχρι **δύο** ατόμων **αυστηρώς.**

Προθεσμία Παράδοσης: Δευτέρα 2 Ιουλίου 2018, θυρίδα διδάσκοντος.