Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://courses.softlab.ntua.gr/progtech/

Προγραμματιστικές Τεχνικές

Το μάθημα θα εξεταστεί φέτος στην κανονική εξεταστική περίοδο σύμφωνα με συνδυασμό των τρόπων εξέτασης Α, Β2γ και Β2δ (βλ. απόφαση της 6ης/2020 συνεδρίασης της Συγκλήτου του ΕΜΠ, 29/5/2020, όπως δημοσιεύθηκε στη Διαύγεια: ΑΔΑ: ΡΟ8Τ46ΨΖΣ4-ΟΒΓ). Με την εξέταση θα ελεγχθεί η επάρκεια των γνώσεων των εξεταζόμενων και θα απονεμηθεί βαθμός απαλλαγής (pass/no-pass).

Το προγραμματιστικό πρόβλημα που περιγράφεται παρακάτω αποτελεί το θέμα της γραπτής εξέτασης στο σπίτι με παράδοση εντός τριών (3) ημερών. Υποβάλλετε τις λύσεις σας στο γνωστό αυτόματο σύστημα ελέγχου `http://grader.softlab.ntua.gr` — αν δεν έχετε κωδικό (`pi19bXYZ` ή `pt20aXYZ`), φροντίστε να αποκτήσετε επικοινωνώντας στο Teams (σε προσωπικό chat) με τον Γιώργο Γκούμα ή την Παρασκευή Τζούβελη εκ των διδασκόντων (μην περιμένετε όμως να σας απαντήσουν εκτός εργασίμων ωρών, φροντίστε να το πράξετε έγκαιρα!).

Άσκηση Δ Δέντρα AVL με μετρητές κόμβων

Προθεσμία υποβολής στον grader: 3/7/2020

Στο σύνδεσμο `https://git.softlab.ntua.gr/pub/avl-tree` μπορείτε να βρείτε μία σχετικά απλή υλοποίηση δέντρων AVL. Στο αρχείο `avltree.hpp` ορίζεται ένα class template για τα δέντρα AVL και υλοποιούνται οι εξής βασικές πράξεις: κατασκευή και καταστροφή δέντρων, προσθήκη, αναζήτηση και αφαίρεση, ενδοδιατεταγμένη (in-order) διάσχιση μέσω iterator.

Ο σκοπός αυτής της άσκησης είναι να τροποποιήσετε το αρχείο `avltree.hpp` και να προσθέσετε στην κλάση `avltree<T>` μία μέθοδο με επικεφαλίδα:

```
1 public:
2   Iterator<T> rank(int n);
```

Προφανώς, μπορείτε να ορίσετε όσες επιπλέον (private) βοηθητικές μεθόδους χρειαστείτε, δεν πρέπει όμως να τροποποιήσετε τα υπάρχοντα περιεχόμενα του αρχείου.

Η μέθοδος `rank` θα πρέπει να επιστρέφει έναν iterator που να δείχνει στον κόμβο με το n-οστό μικρότερο κλειδί που περιέχεται στο δέντρο. Η αρίθμηση πρέπει να αρχίζει από το μηδέν. Δηλαδή, για $n = 0$ θα πρέπει να επιστρέφει τον κομβο με το μικρότερο κλειδί, για $n = 1$ εκείνον με το αμέσως μεγαλύτερο, κ.ο.κ. Αν δεν ισχύει $0 \leq n < t.\text{size}()$, τότε πρέπει να επιστρέφει `end()`.

Σημείωση: Η μέθοδος `rank` θα πρέπει να έχει πολυπλοκότητα $O(\log N)$, όπου N το πλήθος των κόμβων που περιέχονται στο δέντρο. Προσπαθήστε να την υλοποίησετε με ένα "κατέβασμα" του δέντρου, ιδανικά χωρίς αναδρομή, όπως κάνει η μέθοδος `lookup`.

Για να πετύχετε την επιθυμητή πολυπλοκότητα, μπορείτε να προσθέσετε στους κόμβους του δέντρου ένα ακόμα πεδίο με τιμή ακέραιο αριθμό: το μέγεθος (δηλαδή το πλήθος των κόμβων) του υποδέντρου που έχει αυτόν τον κόμβο για ρίζα. Στην περίπτωση αυτή, το πεδίο `the_size` δε χρειάζεται πια. Θα χρειαστεί να τροποποιήσετε κατάλληλα τις μεθόδους `insert` και `remove`, έτσι ώστε να ενημερώνουν κατάλληλα τα μεγέθη των κόμβων. Επίσης, θα πρέπει να τροποποιήσετε κατάλληλα τον κώδικα των μεθόδων που επιτυγχάνουν το ισοζύγισμα του δέντρου.

Μπορείτε να δοκιμάσετε την υλοποίησή σας με προγράμματα όπως το `example.cpp` που δίνεται στο repository της υλοποίησης των δέντρων AVL και με τα test cases που θα βρείτε εκεί. Μπορείτε να προσθέσετε μία εντολή '`k`', παρόμοια με την '`l`', που να εκτυπώνει το κλειδί με τη δοθείσα τάξη (δηλαδή

το n-οστό μικρότερο, όπως είπαμε νωρίτερα). Προφανώς θα χρειαστεί να φτιάξετε και δικά σας test cases που να χρησιμοποιούν αυτή την εντολή.

Προσοχή!

- Το αρχείο που θα ανεβάσετε στον grader θα πρέπει να είναι το τροποποιημένο `avltree.hpp`. Μην πειράξετε οτιδήποτε εκτός από αυτά που ζητάει η άσκηση!
- Φροντίστε να βάλετε στην πρώτη γραμμή ένα **σχόλιο με το ονοματεπώνυμο και τον αριθμό μητρώου σας!**