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 Β Iterators για προθεματική και επιθεματική διάσχιση

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>` τη δυνατότητα προδιατεταγμένης (pre-order) και μεταδιατεταγμένης (post-order) διάσχισης μέσω iterator, επιπλέον της ενδοδιατεταγμένης (in-order) που έχει ήδη υλοποιηθεί. Συγκεκριμένα, ζητείται να προσθέσετε εκτός της κλάσης `avltree<T>` την απαρίθμηση:

```
enum TreeOrder { PREORDER, INORDER, POSTORDER };
```

και εντός της κλάσης `avltree<T>` μία (υπερφορτωμένη) μέθοδο με επικεφαλίδα:

```
public:
  Iterator<T> begin(TreeOrder order);
```

που να επιστρέφει iterator για τη διάσχιση του δέντρου με τη συγκεκριμένη σειρά. Προφανώς, μπορείτε να ορίσετε όσες επιπλέον (private) βοηθητικές μεθόδους χρειαστείτε, δεν πρέπει όμως να τροποποιήσετε τα υπάρχοντα περιεχόμενα του αρχείου, εκτός της υπάρχουσας μεθόδου `begin` και όσων προτείνονται παρακάτω.

Για να υλοποιήσετε τους επιπλέον iterators, μπορείτε να κρατήσετε την κλάση `TreeIteratorImpl` ως βασική κλάση και να ορίσετε τρεις υποκλάσεις αυτής (π.χ., την `TreeInOrderIteratorImpl`, την `TreePreOrderIteratorImpl` και την `TreePostOrderIteratorImpl`) για τους τρεις τύπους iterator. Οι τρεις αυτές κλάσεις θα διαφοροποιούνται μόνο ως προς τη μέθοδο `advance` που θα είναι τετριμμένη στη βασική κλάση `TreeIteratorImpl`. Η υλοποίηση της μεθόδου `begin` θα επιστρέφει iterator με το σωστό τύπο υλοποίησης, ανάλογα με την παράμετρό της, και θα την αρχικοποιεί με το σωστό κόμβο έναρξης της διάσχισης.

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 εντολές 'o' και 'q', παρόμοιες με την 'p', που να εκτυπώνουν τους κόμβους του δέντρου με προδιατεταγμένη (pre-order) και μεταδιατεταγμένη (post-order) σειρά αντίστοιχα, χρησιμοποιώντας τους αντίστοιχους iterators. Προφανώς θα χρειαστεί να φτιάξετε και δικά σας test cases που να χρησιμοποιούν αυτές τις εντολές.

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!**