Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
https://courses.softlab.ntua.gr/pl2/

Γλώσσες Προγραμματισμού II

Αν δεν αναφέρεται διαφορετικά, οι ασκήσεις πρέπει να παραδίδονται στους διδάσκοντες σε ηλεκτρονική μορφή μέσω του συνεργατικού συστήματος ηλεκτρονικής μάθησης `moodle.softlab.ntua.gr`. Η προθεσμία παράδοσης θα τηρείται αυστηρά. Έχετε δικαίωμα να καθυστερήσετε το πολύ μία άσκηση.

Άσκηση 2 Τα δέντρα, πώς να τα διπλώνετε και να τα κλαδεύετε

Προθεσμία παράδοσης: 15/11/2017

Έστω στη γλώσσα Haskell ο τύπος `Tree` του δέντρου με οσαδήποτε παιδιά ανά κόμβο, ο ίδιος που εξετάσαμε στα παραδείγματα με τη διάσχιση DFS/BFS, στις παραδόσεις της 11/10/2017 και 18/10/2017.

```
data Tree a = T a [Tree a]
```

1. Υλοποιήστε την αναδρομική συνάρτηση `foldTree` που "διπλώνει" ένα δέντρο με τον ίδιο τρόπο όπως οι `foldl` και `foldr` διπλώνουν λίστες.

   ```
   foldTree :: (a → [b] → b) → Tree a → b
   ```

   Η πρώτη παράμετρος της `foldTree` είναι μια συνάρτηση `f` τέτοια ώστε `f x bs` είναι το επιθυμητό αποτέλεσμα για ένα δέντρο που περιέχει την τιμή `x` στη ρίζα του και που τα επιθυμητά αποτελέσματα για τα παιδιά του δίνονται στη λίστα `bs`.

2. Στη συνέχεια, χρησιμοποιώντας τη `foldTree`, υλοποιήστε τις παρακάτω συναρτήσεις:

   - `sizeTree t :: Num b ⇒ Tree a → b`:
     το πλήθος των κόμβων του δέντρου `t`.
   - `heightTree t :: (Ord b, Num b) ⇒ Tree a → b`:
     το "ύψος" του δέντρου `t`, δηλαδή το μήκος του μεγαλύτερου μονοπατιού από τη ρίζα σε κάποιο φύλλο (αν η ρίζα δεν έχει παιδιά, η απάντηση πρέπει να είναι 1).
   - `sumTree t :: Num a ⇒ Tree a → a`:
     το άθροισμα των τιμών των κόμβων του δέντρου `t`.
   - `maxTree t :: Ord a ⇒ Tree a → a`:
     η μέγιστη των τιμών των κόμβων του δέντρου `t`.
   - `inTree x t :: Eq a ⇒ a → Tree a → Bool`:
     ελέγχει αν η τιμή `x` βρίσκεται σε κάποιο κόμβο του δέντρου `t`.
   - `nodes t :: Tree a → [a]`:
     μια λίστα που περιέχει τις τιμές όλων των κόμβων του δέντρου `t`.
   - `countTree f t :: (a → Bool) → Tree a → Integer`:
     το πλήθος των κόμβων του δέντρου `t`, οι τιμές των οποίων ικανοποιούν το κατηγόρημα `f`.
   - `leaves t :: Tree a → [a]`:
     μια λίστα που περιέχει τις τιμές όλων των φύλλων του δέντρου `t`.
   - `mapTree f t :: (a → b) → Tree a → Tree b`:
     το δέντρο που προκύπτει αν σε κάθε κόμβο του δέντρου `t` αντικαταστήσουμε την τιμή `x` με `f x`.

Η υλοποίηση των παραπάνω συναρτήσεων πρέπει να γίνει χωρίς τη χρήση αναδρομής με μία και μόνο κλήση στη συνάρτηση `foldTree`, με κατάλληλη παράμετρο. (Για τις περισσότερες από αυτές, η υλοποίηση θα είναι της μίας γραμμής.)

3. Υλοποιήστε (ελεύθερα) τις παρακάτω συναρτήσεις:

   - `trimTree n t :: Int → Tree a → Tree a`:
     το αποτέλεσμα του "κλαδέματος" του δέντρου `t` στο ύψος `n`, δηλαδή το δέντρο `t` αν αφαιρέσουμε όλους τους κόμβους που η απόστασή τους από τη ρίζα δεν είναι μικρότερη του `n`.
   - `path l t :: [Int] → Tree a → a`:
     η τιμή του κόμβου στον οποίο φθάνουμε αν ξεκινήσουμε από τη ρίζα του δέντρου `t` και ακολουθήσουμε το μονοπάτι που ορίζεται από τη λίστα `l`. Για παράδειγμα, η λίστα `[0,1]` μας οδηγεί στο πρώτο παιδί της ρίζας (0) και από εκεί στο δεύτερο παιδί του (1). Θεωρήστε δεδομένο ότι θα είναι δυνατό να ακολουθήσετε το μονοπάτι `l` μέσα στο δέντρο `t`, διαφορετικά η συνάρτησή σας μπορεί να προκαλεί σφάλμα εκτέλεσης.

Άσκηση 3 Το πουλί, το άπειρο δέντρο και οι γρήγορες δοκιμές

Προθεσμία παράδοσης: 15/11/2017

Έστω στη γλώσσα Haskell ο ίδιος τύπος `Tree` και οι συναρτήσεις που αναπτύξατε για την άσκηση 2.

1. Υλοποιήστε μία γεννήτρια αυθαίρετων δέντρων που να επιτρέπει τον αυτόματο έλεγχο ιδιοτήτων για δέντρα με το εργαλείο `QuickCheck`. Συγκεκριμένα, υλοποιήστε ένα `instance` της μορφής:

   ```
   instance Arbitrary a ⇒ Arbitrary (Tree a)
   ```

   Επειδή ο τύπος `Tree` είναι αναδρομικός, φροντίστε να κάνετε χρήση του μέτρου στη γεννήτρια (`sized`, `resize`), έτσι ώστε η γεννήτρια να τερματίζει. Επίσης, φροντίστε ώστε τα δέντρα που γεννάτε να μην είναι πάντοτε είτε πολύ "πλατειά" είτε πολύ "στενά".

2. Χρησιμοποιήστε τη γεννήτρια για να ελέγξετε με το `QuickCheck` τις παρακάτω ιδιότητες:

   - Το ύψος κάθε δέντρου (`heightTree`) είναι θετικός φυσικός αριθμός που δεν υπερβαίνει το μέγεθός του (`sizeTree`).
   - Η μέγιστη τιμή ενός δέντρου (`maxTree`) ανήκει (`inTree`) στο δέντρο.
   - Όλοι οι κόμβοι (`nodes`) ενός δέντρου ανήκουν στο δέντρο.
   - Το πλήθος των κόμβων ενός δέντρου που πληρούν μια ιδιότητα (`countTree`) είναι φυσικός αριθμός που δεν υπερβαίνει το μέγεθος του δέντρου.
   - Το πλήθος των κόμβων ενός δέντρου είναι ίσο με το μέγεθός του. Επίσης, το πλήθος των φύλλων είναι μικρότερο του πλήθους των κόμβων, εκτός αν και τα δύο ισούνται με 1.
   - Η συνάρτηση `mapTree` διατηρεί το μέγεθος και το ύψος του δέντρου.
   - Αν η τιμή `n` ανήκει στους κόμβους ενός δέντρου `t`, τότε η τιμή `f n` ανήκει στο δέντρο `mapTree f t`.
   - Για κάθε συνάρτηση `g` ∈ { `nodes`, `leaves` } ισχύει: `map f . g == g . mapTree f`.

3. Ορίστε το δέντρο του Bird [Hin09], ένα άπειρο δυαδικό δέντρο με τύπο:

   ```
   bird :: Tree Rational
   ```

   Ο ορισμός σας, χρησιμοποιώντας κάποιες από τις συναρτήσεις που ορίσατε στην άσκηση 2, δεν έχει λόγο να υπερβαίνει τις 3 γραμμές...

4. Χρησιμοποιήστε το `QuickCheck` για να ελέγξετε τις παρακάτω ιδιότητες του δέντρου του Bird:

   - Ακολουθώντας ένα μονοπάτι μήκους `n` είτε στο `bird` είτε στο `trimTree n bird` οδηγούμαστε στον ίδιο κόμβο.
   - Ακολουθώντας ένα μονοπάτι από τη ρίζα που κάνει "ζιγκ-ζάγκ", δεξιά-αριστερά, οδηγούμαστε διαδοχικά σε όλους τους θετικούς φυσικούς αριθμούς.
   - Ακολουθώντας από τη ρίζα το αριστερότερο δυνατό μονοπάτι, οδηγούμαστε σε κόμβους που οι παρονομαστές τους ορίζουν την ακολουθία Fibonacci.
   - Κάθε θετικός ρητός αριθμός εμφανίζεται σε κάποια θέση στο δέντρο του Bird. Για να ελέγξετε αυτή την ιδιότητα, ίσως θα χρειαστείτε να γράψετε μία συνάρτηση `findBird q`, η οποία να επιστρέφει το μονοπάτι που από τη ρίζα οδηγεί στο ρητό αριθμό `q`.

Αναφορές

[Hin09] Ralf Hinze. The Bird tree. *Journal of Functional Programming*, 19(5):491–508, September 2009. Διαθέσιμο από: http://www.cs.ox.ac.uk/ralf.hinze/publications/Bird.pdf.