Segment Trees Haskell Implementation

Jiří Škrobánek[1]

May 14, 2019, Prague

Introduction

This document serves as a reference for `SegmentTree` Haskell module. The module provides an implementation of segment trees usable with any type of keys which implements `Ord` and can aggregate any associative function of values.
All editing operations take $\mathcal{O}(\log n)$ time on a tree with n elements as well as range queries. Insertion and deletion is implemented using AVL-tree balancing algorithms yielding the same $\mathcal{O}(\log n)$ time complexity.

Example Usage

In this example we build a tree on pairs of integers, to do this we must first instantiate the aggregation, in this case aggregation is simply addition.
Firstly we build a tree on integers from 1 to 10. We update the value at 5 by adding 2 to it, lastly we ask for the sum of interval 2 to 8.

```
import SegmentTree

instance Aggregable Integer where
    aggregate Nothing x = x
    aggregate x Nothing = x
    aggregate (Just x) (Just y) = Just (x+y)

x = segAVLBuildFromList [(x,x) | x <- [1..10]]
y = segSetValue x 5 2 (\x y -> x + y)
z = segAVLRange y 2 8
```

Complete List of Functionality

```
class Aggregable v where
aggregate :: (Maybe v) -> (Maybe v) -> (Maybe v)
```

Values must be an instance of this class.

```
data SegAVL k v
| Node {key::k, value::v, lsub::SegAVL k v, rsub::SegAVL k v, height::Int, agg::Maybe v}
| Empty
```

Functions

- `segAVLBuildFromList :: (Ord k, Aggregable v) => [(k,v)] -> SegAVL k v`

  This function creates the segment tree from a sorted list of key-value pairs.

- `segAVLInsertAndBalance :: (Ord k, Aggregable v) => SegAVL k v->k->v-> SegAVL k v`

  This function inserts the key-value pait into the tree, overwriting old values.

- `segAVLSetValue :: (Ord k, Aggregable v) => SegAVL k v->k->v->(v->v->v)->SegAVL k v`

  Similar to the previous function, but takes an extra argument which handles the replacing of old value.

- `segAVLDelete :: (Ord k, Aggregable v) => SegAVL k v -> k -> SegAVL k v`

[1]Faculty of Mathematics and Physics, Charles University, jiri@skrobanek.cz

This function removes the given key from the tree.

- `segAVLRange :: (Ord k, Aggregable v) => SegAVL k v -> k -> k -> Maybe v`

  This function return the aggregated value from interval between the second and third parameter.

- `segAVLFind :: (Ord k, Aggregable v) => SegAVL k v -> k -> Maybe v`

  Returns the value associated with given key.

- `segAVLMember :: (Ord k, Aggregable v) => SegAVL k v -> k -> Bool`

  Return a Boolean determining wheter the key is in the tree.

- `segAVLToList :: (Ord k, Aggregable v) => SegAVL k v -> [(k,v)]`

  Produces an ordered set of key-value pairs from the tree.

- `segAVLGetMin :: SegAVL k v -> Maybe (k,v)`

  Returns the minimum key-value pair present in the tree.

- `segAVLGetMax :: SegAVL k v -> Maybe (k,v)`

  Returns the maximum key-value pair present in the tree.

- `segAVLLowerBound :: (Ord k) => SegAVL k v -> k -> Maybe (k,v)`

  Return the key-value pair where the key is the smallest not lesser then the key provided.

- `segAVLUpperBound :: (Ord k) => SegAVL k v -> k -> Maybe (k,v)`

  Return the key-value pair where the key is the greatest lesser then the key provided.