WUOLAH

ex.pdf

Anónimo

Estructuras de Datos

2º Grado en Ingeniería del Software

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

Exam Statement

Important At the end of the exam you should upload only in an **uncompressed** way files Seq.hs, LinkedSeq.java, BinaryTree.hs, BinaryTree.java, DiGraphDftTimer.hs, DiGraphDftTimer.java, WBinTree.hs and WBinTree.java with your solutions. Make sure that all the files you have uploaded are really those corresponding to your solution, that they all contain in the initial comments your name and identity number/passport and that they compile without errors. Only files submitted without compilation errors will be graded.

Exercise 1

Add a single-digit number to a sequence representing a number.

Haskell (1.25 points). File Seq.hs

Given a single-digit number k and a sequence whose nodes store digits of a non-negative number, add k to the sequence (working directly with the sequence and without inverting it).

The datatype for representing a sequence is the following one:

```
data Seq a = Empty | Node a (Seq a) deriving (Eq, Show)
```

For example, consider the sequence `Node 9 (Node 9 (Node 9 (Node 3 Empty)))` which represents the number 9993. Adding the single-digit number 7 to it should result in the sequence `Node 1 (Node 0 (Node 0 (Node 0 (Node 0 Empty))))` which corresponds to the number 10000.

The function you should define is the following one:

```
addSingleDigit :: (Integral a) => a -> Seq a -> Seq a
```

defined in file `Seq.hs`. File `SeqDemo.hs` includes a test function.

Java (1.25 points). File LinkedSeq.java

Given a single-digit number k and a linked sequence whose nodes store digits of a non-negative number, add k to the sequence (working directly with the sequence and without inverting it).

The class for representing a linked sequence is `LinkedSeq<T>`, provided with the codes for the exam.

For example, consider the linked sequence `9 -> 9 -> 9 -> 3 -> null` which represents the number 9993. Adding the single-digit number 7 to it should modify the linked sequence so that it becomes `1 -> 0 -> 0 -> 0 -> 0 -> null`, which corresponds to the number 10000.

The method you should define is the following one:

```
public static void addSingleDigit(int d, LinkedSeq<Integer> linkedSeq)
```

in class `LinkedSeq` in package `exercises`. Class `LinkedSeqDemo` includes a test.

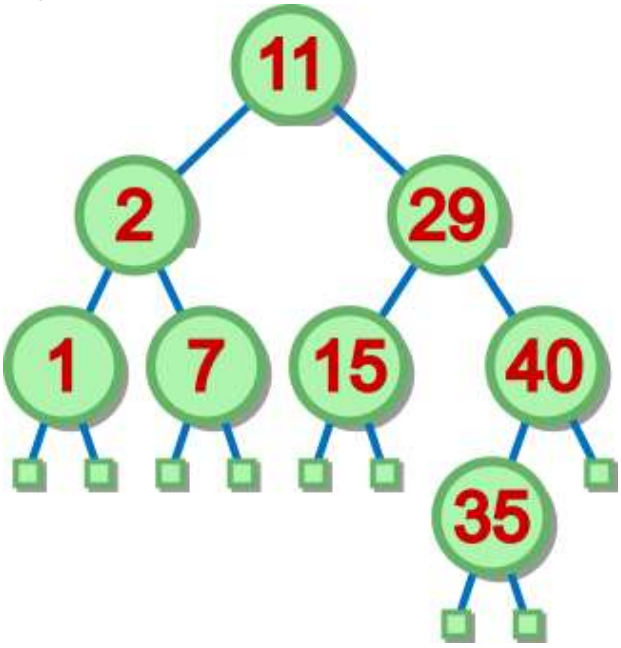
Exercise 2

Number of subtrees with all nodes in a range of values.

Let `t` be a binary search tree (BST) and let `min` and `max` be two values. You should compute how many subtrees in `t` have all values in their nodes within the interval defined by `[min,max]`. ***Your solution should have linear complexity with respect to the number of nodes in the original tree.*** For this purpose, we recommend using a bottom-up algorithm, transfering information from children to parents. In this way, we can say that a tree has all its nodes within the interval if its root is within the interval and, at the same time, both its left and right children have all their nodes within the interval.

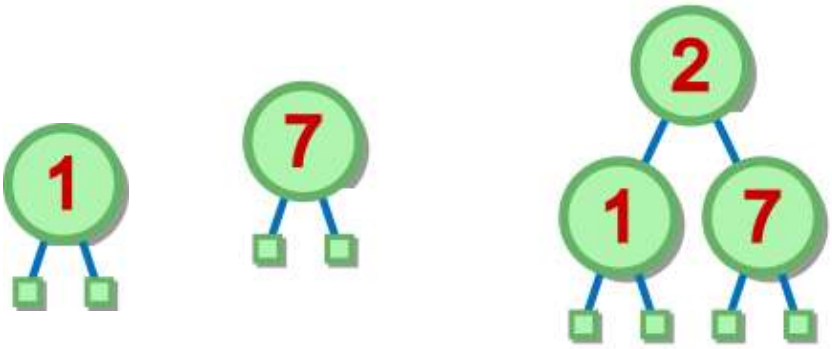
Original tree

Original tree



For instance, the tree in the figure above has 3 subtrees within the interval [0, 10]. These trees are shown below:

Subtree 1 Subtree 2 ***Subtree 3***



Haskell(1.25 points). File BinaryTree.hs - Function subTreesInRange

Complete the implementation of the following function in module `BinaryTree`

```
subTreesInRange :: (Ord a) => BinaryTree a -> a -> a -> Integer
```

taking a binary search tree (BST) and two values (min and max) and returning the number of subtrees having all their nodes within interval [min,max]. You can assume that min < max. File SubTreeDemo.hs includes a test for this function.

Java (1.25 points). File BinaryTree.java - Method subTreesInRange

Implement the following method in class `BinaryTree<T>` in package `exercises`

```
public int subTreesInRange(T min, T max)
```

which should return the number of subtrees having all their nodes within interval [min,max].

Main method in class `BinaryTreeDemo` can be used to test your solution.

Exercise 3

academias
UNITEC

28 años formando ingenieros

ESPECIALIZADAS en INGENIERÍAS

INTENSIVOS

| Convocatoria ENERO - FEBRERO |

Especial atención a la RESOLUCIÓN de PROBLEMAS DE EXÁMEN

- Profesorado EXPERTO
- Alto nº de APROBADOS
- Apoyo individualizado

Presencial u online ¡tú eliges!

Contacta con nosotros

C/ Eolo 3. 29010 Málaga
C/ Teseo 9. 29010 Málaga
95 2345678
info@academiasunitec.com
@academias_unitec_
academiasunitecmalaga

www.academiasunitec.com

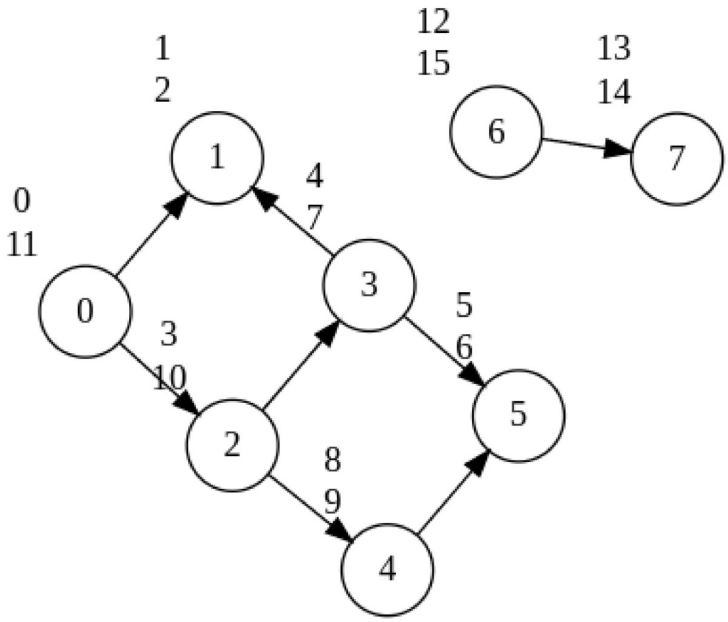
Arrival and departure times of vertices in Depth-First Traversal.

Haskell (1.25 points). File DiGraphDftTimer.hs

Given a directed graph, find *arrival and departure times* of its vertices when performing a Depth-First Traversal (DFT). *Arrival time* is the time at which the vertex was explored for the first time during the DFT, and *departure time* is the time at which we have explored all the successors of the vertex and we are ready to backtrack to another vertex. Times are defined as a sequence of integers (0 for first visited node, 1 for second one, and so on).

For instance, the graph in the figure below is annotated with arrival and departure times on left top of each node for a possible DFT starting from vertex 0.

Graph annotated with arrival and departure times



The function you should define is the following one:

```
diGraphDftTimer :: (Ord v) => DiGraph v -> (Dictionary v Int, Dictionary v Int)functions
```

returning a dictionary assigning each vertex to its arrival time and another dictionary with departure times.

Module `DiGraphDftTimerDemo` includes a test for your solution.

Java (1.25 points). File DiGraphDftTimer.java

Given a directed graph, find *arrival* and *departure* times of its vertices when performing a Depth-First Traversal (DFT). *Arrival time* is the time at which the vertex is explored for the first time during the DFT, and *departure time* is the time at which we have explored all the successors of the vertex and we are ready to backtrack to another vertex. Times are defined as a sequence of integers (0 for first visited node, 1 for second one, and so on).

The class you should complete is `DiGraphDftTimer<V>` in package `exercises`, so that it fills up dictionary `arrivalD` in order to assign each vertex to its arrival time and dictionary `departureD` with departure times.

Class `DiGraphDftTimerDemo` includes a test for your solution.

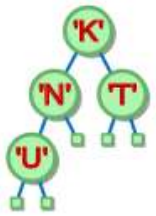
Exercise 4 (Only for students without continuous assessment)

Weight-balanced binary trees

Let the *weight* of a tree be its total number of nodes. A binary tree (not necessarily a binary search tree) is weight-balanced if it satisfies the following invariant: the weigth of left subtree is either exactly the same as the weight of the right subtree, or one element larger.

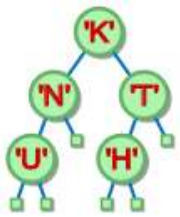
For example, the tree in the figure below is a weight-balanced tree:

Tree before



Now, if we insert element `H`, as we need to adhere to the invariant, we will obtain the following weight-balanced tree:

Tree after



Haskell (0.50 + 0.75 points). File WBinTree.hs

Let us represent weight-balanced binary trees with type `WBinTree` in module `WBinTree`. Complete the implementation of following functions:

```
isWeightBalanced :: WBinTree a -> Bool
```

which should determine if a binary tree is weight-balanced.

```
insert :: a -> WBinTree a -> WBinTree a
```

which should insert a new element in a weight-balanced binary tree, enforcing the invariant so that the output is also weight-balanced.

Module `WBinTreeDemo` includes a test for these functions.

Java (0.50 + 0.75 points). File WBinTree.java

Let us represent weight-balanced binary trees with class `WBinTree` in package `exercises`. Complete the implementation of following methods:

```
public boolean isWeightBalanced()
```

which should determine if a binary tree is weight-balanced.

```
public void insert(T x)
```

which should insert a new element in a weight-balanced binary tree, enforcing the invariant so that the resulting tree is also weight-balanced.

Class `WBinTreeDemo` includes a test for these functions.