Ciências ULisboa | Informática

VVS (2019/2020)
Assignment 1

1 Introduction

This assignment goal is to prepare a set of unit tests to verify an implementation of a Trie data structure, a tree for fast insertion and retrieval of key-value pairs. In this context, the SUT to analyse is a variant using ternary trees.

Please read section 5.2 from Algorithms, 4th Edition by Robert Sedgewick and Kevin Wayne, for details about this data structure[1]. Import maven project `vvs_assignment1` to access the source code implementing Tries.

2 Requirements

The tests you should include must satisfy the requirements produced by the following criteria:

1. Instruction Coverage for all public methods.
2. Edge Coverage and Prime Path Coverage for method `longestPrefixOf`
3. All-Uses Coverage for method `longestPrefixOf`
4. Select and apply one Logic-based test coverage for method `longestPrefixOf`, justify your option

For method `put`, include a test set based of Input State Partitioning, namely Base Choice Coverage, using the following characteristics:

1. Trie already includes the new key
2. Trie already includes some new key prefix
3. Trie is empty

Verify your test set via program mutation using PIT. Write a report comparing the mutation coverage achieved by each criteria. Then, if needed, add new tests to improve mutation coverage.

Use JUnit QuickCheck to create a Trie random generator. Include a `equals()` and a `delete()` method in the class. Test the following properties

1. The order of insertion of different keys does not change the final tree value
2. If you remove all keys from a tree, the tree must be empty
3. Given a tree, inserting and then removing the same key value will not change its initial value

[1] There's a video explanation by the authors at `https://www.youtube.com/watch?v=CIGyewO7868`

4. Selecting a stricter prefix `keysWithPrefix` returns a strict subset result. Eg, `keysWithPrefix("sub")` $\subseteq$ `keysWithPrefix("su")`

For each coverage criteria paradigm, include its test cases in different classes or even different packages (if you have several classes per paradigm), all of them properly named. There's no problem if you repeat test cases between packages.

For each unit test include appropriate Java comments that detail what test requirements it satisfies.

3 Deliver

Upload a zip of the group's work into the course's moodle webpage until 23:59 of May 9. The zip file should be named `vvs01_XX.zip` where `XX` is your group's number.

The zip must include:

- the maven project (with just the source code and `pom.xml`, no binaries or Eclipse metadata)
- a pdf report with your analysis of the several coverage criteria