Writeup for second project of
CMSC 420: "Data Structures"
Section 0102 , Summer 2017

Theme: Binary Patricia Tries

Handout date: Wednesday, 06-21-2017
On-time deadline: 06-30-2017, 11:59pm
Late deadline (30% penalty): 07-02, 11:59pm

1 Overview

In this programming project, you will implement **Binary Patricia Tries**. Binary Patricia Tries are Patricia Tries over the binary alphabet $\{0, 1\}$. The goal of the project is to construct and maintain Patricia tries with simple binary strings as input. All programming will be done in Java, and you will be graded automatically by the CS department's submit server. This write-up contains guidelines about what needs to be implemented as well as instructions on submitting your project.

2 Prerequisites

We expect that you have familiarity with both Tries and Patricia Tries. Review the class recordings and slides if necessary to make sure you are up to speed with these two excellent data structures.

3 Project Description

3.1 Binary Patricia Tries

Your project involves implementing insertion, deletion, lookup, traversal algorithms for Binary Patricia tries (along with a few additional methods). Each node of your trie will be a record containing only the following information:

1. The index of the character in the input key to test at that node.
2. The left and right child of the node, one for each of the 2 possible next values (0 or 1) in the key, respectively.
3. A reference to the key itself, if the key ends at the index.

3.2 Implementation

Everything you need is available on our common git repository. You will need to fill-in the implementation of the class `BinaryPatriciaTrie`. The class comes with sufficient documentation that you will be able to find under the `doc` subdirectory, so that you can have a full view of the functionality exposed by the class's `public` interface.

Please note that, in its default state, the file `BinaryPatriciaTrie.java` will have a `package` declaration which will probably not make sense for your directory structure. Feel free to comment out the `package` declaration at the top of the file, since **your file will need to be in the default package for the unit tests to access the methods**. Any other classes, enums, clients, test suites, etc that you want to compile for your project to run can lie within any package hierarchy they choose.

Some points about the implementation:

- The input type for a key is a `String`. Specifically, in our case, it will be a **binary string**, i.e it will contain only '0's and '1's. Example binary strings are "0101010", "1101" and "0". Since the input type is known at compile-time, this class, unlike the first project's `ThreadedAVLTree` class, does not have to be a generic! Don't go searching for any unknown types `T`! You won't find any :)
- Recall that it is **absolutely possible** that the trie may contain a key that is a prefix of other keys! In this case, the nodes that contain these "prefix keys" are non-leaf nodes. Do not assume that keys are located only in the leaves of the trie!
- For this project, we assume that there are **no duplicate keys in your data structure**. This means that, in our unit tests, whenever we delete a key from your tree, **we expect it to no longer be found in the tree**. And we **test for this!**

To pass most of our unit tests, you will need to implement **all** of `BinaryPatriciaTrie`'s `public` methods. This is because a single unit test might use **many methods** of the class's interface in order to make sure that you are not having any **hidden errors in the structure of your trie**. Such errors will invariably make your code fail the harder tests, which tend to stress test a method or two each! Note that, if the Java runtime fails to find a particular method implemented, it will throw a relevant exception and the corresponding unit test will, naturally, fail.

4 Submission / Grading

Projects in this class are different from your typical 131/2 projects in that we do not maintain a CVS repository for you or us. This means that you can no longer use the Eclipse Course

Management Plugin to submit your project on the submit server. This turns out to be a good thing, since it frees you from the need to use Eclipse (or any IDE for that matter) if you don't want to. To submit your project, make a .zip file of your **ENTIRE PROJECT FOLDER** and upload it on the submit server. See figure 1 for the part of the online interface where you can upload your project.

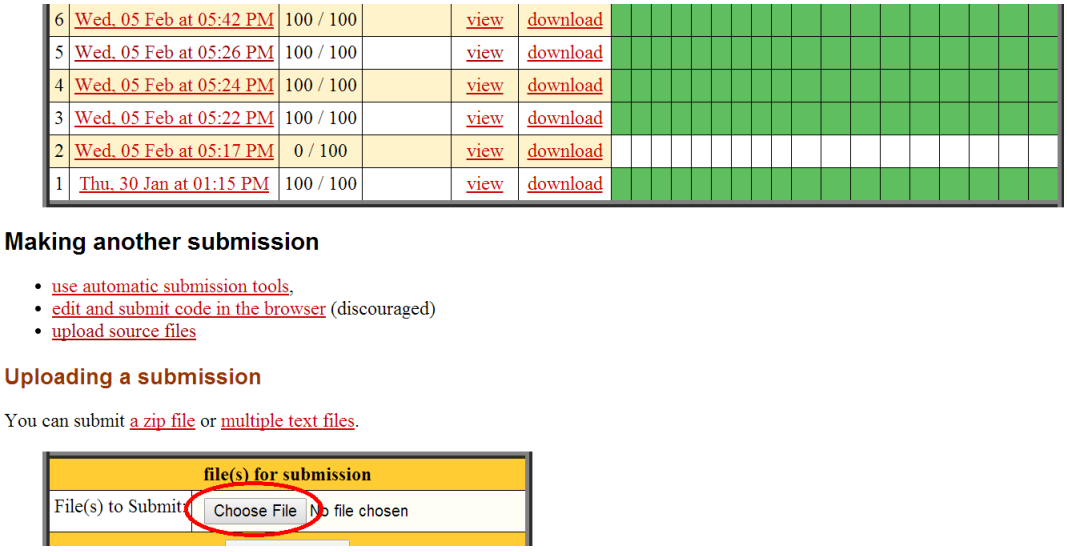

Figure 1: Uploading your project on the submit server.

All tests are release tests, and you can submit **up to 5 times** every 24 hours. We urge you to **unit-test your code thoroughly before submitting**: treat every token like a gold bar that is not to be wasted! We will **not** share the source code of the unit tests with you until after the late deadline for the project! We maintain your **highest-scoring submission** for grading purposes.

For the late deadline, we take 30% off your maximum possible score. This means that, if you submit late, passing all the unit tests and implementing inorder traversal **as discussed** will give you 70% of the total grade.

Finally, this project is a Patricia Trie project, not a Trie project. **NO CLASSIC TRIE IMPLEMENTATION WILL BE ACCEPTED, EVEN IF IT PASSES ALL UNIT TESTS!** We will be observing your source code after submission to make sure of this! We have already completed lecture exercises that involve source code of tries, and that code is in the slides; if you feel you want to adapt these methods for your **Patricia** trie, that is obviously fine; **but patching up a classic trie implementation is not**. In fact, depending on how you implement your various methods, a classic trie will fail many of our tests because of `StackOverflowException`s! The JVM is not very good with stack management =/

Good luck!


PICTURE UNRELATED