Wikigraph

In this assignment we will implement an algorithm and all the tools to compute the distance between wikipedia articles.

Download the handout here.

Introduction

But what is the **distance** between two pages? We can consider each Wikipedia article (represented by its id) to be a node, of the graph. The links from one page to another one can be interpreted as the edges of this directed graph. Therefore the distance between page *A* and page *B* is the number of links that a user needs to click on to reach page *B* from page *A*.

Overview

The first part of the assignment consists in finishing the implementation of the WikiResult[A] type in the WikiResult.scala file. This type used to describe the result of an asynchronous computation which will produce a value of type A or fail with an error of type WikiError.

Note however that this type is based on Future[Either[Seq[WikiError], A]] which means that you have to take into consideration two kinds of errors:

- *domain errors* which are represented by the Seq[WikiError] type. The future will be a Success(Left(error: Seq[WikiError]))
- *system failure* which are represented by the type WikiException (which is a subclass of Throwable). The future will be a Failure(failure: WikiException)

The Left variant contains a Seq because methods such as zip and traverse collect all the errors from the multiple WikiResult which they combine.

To complete this part you will need to use the common operators on Either and Future such as map and flatMap as well as combinators on Future such as traverse.

The second part of the assignment uses what was built in the first one. In particular, you are asked to implement three functions which consume and produce WikiResult in the Wikigraph.scala file. Note that client: Wikipedia is in scope as argument of the Wikigraph constructor. This object allows you to query the Wikipedia dataset using three simple methods.

The implementation of breadthFirstSearch requires some more details. This method uses a *breadth first search* (BFS) on the graph of articles to compute the distance between two pages. The interaction with the article store will be asynchronous and concurrent.

BFS Algorithm recap

The skeleton of the recursive method iter is provided to you. To fill in the rest, remember that BFS iteration first explores all the neighbors of a node and after that, it moves on to the neighbors of each neighbor.

To do so the algorithm relies on a set of visited nodes to remember which nodes where already explored and therefore avoid being stuck in infinite loops around graph cycles (for example A -> B -> C -> A). It also maintains a (priority) queue which contains the nodes to visit next. In this case, we modified this queue to contain not only the ID of the next nodes to analyze, but also the distance of the node from the start of the search. Finally, the implementation of breadthFirstSearch in this assignment also receives a maxDepth argument which describes the threshold to respect for the exploration.

The algorithm proceeds as follows:

- When no node is left to visit or when we exceed maxDepth, the search is considered failed and completes with a None
- Otherwise, we retrieve the next node to visit (and its distance from the start of the search) and its neighbors. If the destination is amongst the neighbors, our search is complete, and we can return the distance. Otherwise, we add the neighbors to the queue with the associated distance and we perform a recursive call.

Download the dataset

To be able to test your code without Internet access, we have assembled for you a small dataset. This small graph contains all the pages that are at distance 2 from the Scala (programming language) article. The dataset might be incorrect or out of date and should be used only for testing.

You can download the file from here

Please put the file inside a directory named `enwiki-dataset`, in the project root directory. The path to the dataset (relative to the project root directory) should be `enwiki-dataset/smallwiki.db` so that it matches the filename used in `src/main/resources/application.conf`.

Open the Project

Download the handout archive, extract it somewhere on your filesystem, and open the directory wikigraph in your code editor.

In case you use IntelliJ, go to menu “File”, click “Open…”, find the directory “wikigraph” where you extracted the archive, and click OK. IntelliJ should open the project and analyze its structure in the background.

In case you use Metals with VS Code, go to menu “File”, click “Open Folder…”, select the directory “wikigraph” where you extracted the archive and click OK. Metals should show a

notification saying “New sbt workspace detected, would you like to import the build?”, click “Import build”.

You will work on two files:

- src/main/scala/wikigraph/WikiResult.scala
- src/main/scala/wikigraph/Wikigraph.scala

Fill in the Blanks

As you can see, the project is already partially implemented. However, there are still unimplemented parts.

In Scala, we use the symbol ??? as a placeholder for an unimplemented part. So, what you have to do is to replace the occurrences of ??? with proper implementations. Note: don't change the existing definitions, only replace the ??? with proper expressions. If you change the signature of a method, the grader might not work anymore and you will fail the assignment.

Run the Tests

At any point, you can run the tests to follow your progress.

With IntelliJ, click on the green arrow in the left margin, in front of the class definition WikigraphSuite, in the file src/test/scala/wikigraph/WikigraphSuite.scala.

With Metals and VS Code, open the file src/test/scala/wikigraph/WikigraphSuite.scala and click “test” just above the class definition WikigraphSuite.

Run the Program

While you implement the different parts of the assignment, you can experiment with them by modifying the file src/main/scala/wikigraph/Main.scala.

With IntelliJ, click on the green arrow in the left margin, in front of the definition @main def main(): Unit and click on “Run 'Main'”.

With Metals and VS Code, open the file src/main/scala/wikigraph/Main.scala and click “run” above the definition @main def main(): Unit.