Department of Computing

COMP2010/COMP6011 Assignment One Specification

Due date: Week 9 — May 4, 2020.

Assignment marks: 15% of overall unit marks.

Objective: To gain experience in programming with tree data structures; to practise building larger applications from a variety of data structures; to adapt basic tree and sorting algorithms to achieve the best performance possible within the constraints of the given data structures.

Please note: This assignment specification aims to provide as complete a description of this assessment task as possible. However, as with any specification, there will always be things we should have said that we have left out and areas in which we could have done a better job of explanation. As a result, you are strongly encouraged to ask any questions of clarification you might have, either by raising them during a lecture chat or by posting them on the iLearn discussion forum devoted to this assignment.

Page Rank for ranking pages in a Web shaped like a tree

Before Google search, there were a number of searching tools around, but they all suffered from a big problem: their resulting listing of webpages tended to be irritatingly useless because usually they didn't bear much relevance to the original search term. What these search algorithms didn't do was to take account of the "importance" of any particular page. One of the innovations of the original Page Rank algorithm used in Google Search was to factor in a definition of importance. Whether or not you agree with the definition, it certainly makes a tremendous improvement to the quality of the webpages found.

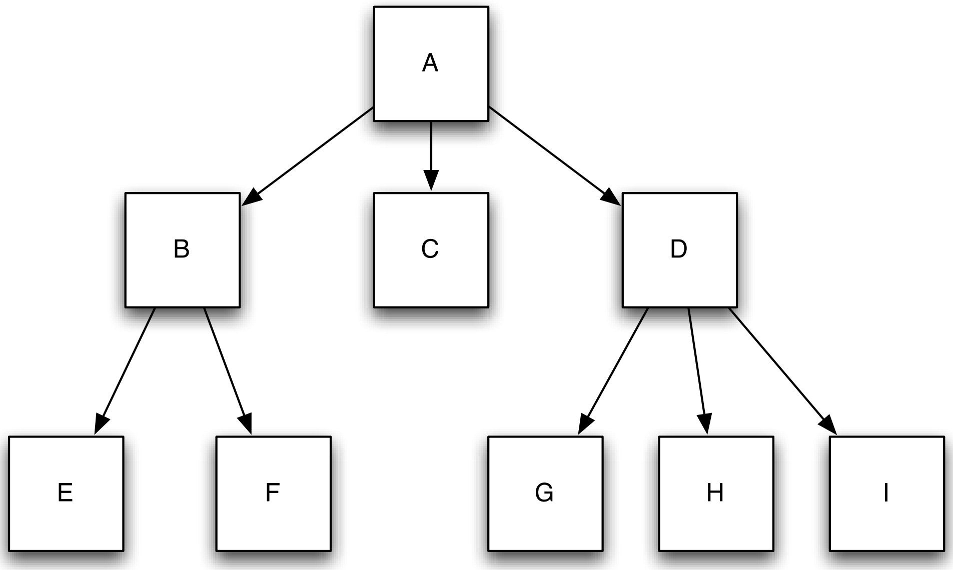
In this assignment you'll learn how to program Page Rank for a very simplistic view of the World Wide Web.

A simple model of the World Wide Web

The Web is very large and complicated, but at its most basic it can be viewed as a huge collection of webpages with links between them. With this view the Web looks like a graph, where nodes are the pages and arcs between nodes are defined by links between pages. In this assignment we'll make a significant simplification, namely that the resulting graph is actually a tree. Of course the real Web is nothing like a tree, but this will help us to get started!

Below is a diagram of the World Wide Web in its early days: it has 9 webpages called A, B,..I. Pages B, C and D are pointed to from A (i.e. on A there are links to these three pages); pages G, H and I are pointed to from D, and so on. Pages C, E, F. G, H and I do not point to any other pages.

Now imagine a person browsing a Web of this shape. He/she might start on Page A and then, after reading a bit, jump randomly to one of the three pages it refers to (namely B, C, D). Sound familiar? At any rate this browsing person will continue reading and probably jump again randomly if there is anywhere to go to. We'll assume (unrealistically, but just to simplify the problem) that after reaching a leaf the person stops browsing altogether.



The idea behind Page Rank is that a page's importance increases if other pages refer to it (via linking), and the random browser splits arbitrarily the importance of a reference on a page between the other references. Thus since page A has three references the importance conferred to the children (by virtue of being referenced to) is split evenly between the references. However the importance of the references on page B is only split in two (because there are only two references in total). This is consistent with the view of the random browser described above, because she/he will continue browsing with interest if there are links to follow, but randomly select any reference if there are a number to choose from.

Page Rank actually attempts to put a number on "importance" of a page by taking the shape of the World Wide Web (modelled by a tree in this assignment) into account. Importance increases with a reference, but that increase is reduced if the page making the references also refers to many other pages.

Of course the situation is a little more complicated because some pages should start out with a particular level of importance. For example pages created by he Government's Bureau of Statistics are normally regarded as being more important than a statistics page created by someone just for fun.

Page Rank uses the following formula to compute the importance of a node X which is the child of node Y in the tree:

Importance of X = Base importance of X + (Importance of parent Y)/Number of references on Y.

Here the Base importance is assumed to be some given value for each page, and the Number of references is the total number of children of Y considered as a node in the tree. For example, the Number of references on A is 3, but 2 on B. Observe that once the Importance of the root node is known (A in the diagram) then all the other importances can be calculated by descending into the tree. Note also that the importance of the root node is simply its base importance.

The Java model of the tree shaped Web.

In this assignment we'll regard a model of the World Wide Web as a tree t, such as in the diagram above. Each WebPage has title together with a Vector of links to other WebPages it references. For example in the diagram above the root of the tree t has WebPage with title "A" together with references to WebPages with titles "B", "C" and "D".

Notice that the nodes (WebPages) in a tree can have an arbitrary number of children: A for example has three children, whereas B has two children and C has none. In the AssignmentOne source files we model a WebPage as a structure which has two data fields: `title` (which is a String) and `links` (which is a Vector of WebPages). This allows us to build a model of a tree where each node has an arbitrary number of children. In particular the links field can grow or shrink depending on whether pages are added or deleted.

Importances and Base Importances

At the "Centillion" web-browser headquarters there is a database of WebPages together with their base importance.

```
Map<WebPage, Double> baseImportance= new HashMap<WebPage, Double>();
```

Conceptually this provides a map between addresses (i.e. WebPages) and their base importances.

To store the rankings taking the shape of the Web into account we need a function which takes a tree t and the baseImportances and computes the mapping:

```
Map<WebPage, Double> importance= new HashMap<WebPage, Double>();
```

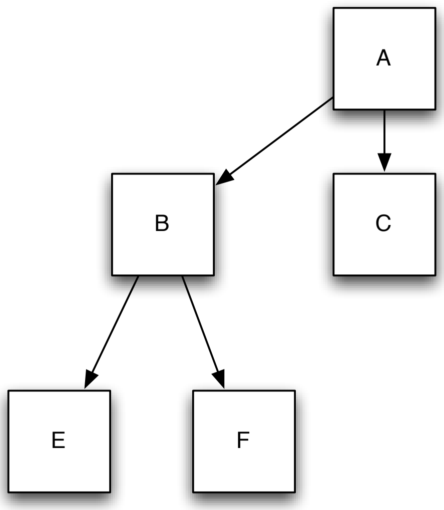
according to the formula given above.

In the example above, where all base ranks are set to 1, the final ranking gives importance 1 to A, but B, C, D an importance of 1.33 each; E, F get an importance of 1.665, and G, H, I an importance of 1.443.

This means that E and F are considered to be the most important pages, followed by G, H and I ; then B, C D and finally A. In a ranking algorithm that prints out the names of the pages according to this importance this means that E and F would appear at the top of the list.

A Dynamic Web

Consider now the case where one of the pages is deleted, say D. The tree t now looks like this:



Of course the importance ranking will now change because the shape of the Web has changed. This explains why a search last year would rank pages differently to the same search performed this year.

Your task

In this assignment, your task is to complete the programs in the Assignment12020.zip which provide the beginnings of the functionality for computing the importance ranking in a simple model of the World Wide Web envisaged as a tree. You are also asked to implement a function for removing a node in the tree and to count the number of nodes in the tree.

Students who wish to complete the assignment at the CR/D/HD level will be asked to provide further functionality.

The 15% credit for this assignment will be awarded entirely on the basis of automarking relative to a set of JUnit tests. A sample of these tests are provided in the source files as a benchmark. Please note that the sample tests assume the given input tree. **When your programs are tested by the auto-tester a different input tree will be used.** You are strongly advised to experiment with different trees to ensure that your programs are correct. This will help you understand the specifications.

What you have to do

Download the Assignment12020.zip file from iLearn, and install it in Eclipse.

Inside the file WorldWideWeb.java there are several stubs for functions. The basic assignment asks for implementations of the following three functions inside the class WorldWideWeb:

```
int numNodes(WebPage t);
```

```
WebPage removeNode(WebPage t, String k);
```

```
void calculateImportances(WebPage t)
```

Completing successfully implementations for the above mentioned methods is sufficient to achieve a Pass level in this assignment.

```
public WebPage changeReference(WebPage r, WebPage q, WebPage s);
// Precondition: WebPage r in the current tree references WebPage q        (i.e. r.links contains q)
// Postcondition: WebPage r removes its reference to q; WebPage s adds a reference to q; returns
updated reference to s
```

Completing successfully implementations for all the above mentioned methods is sufficient to achieve a Credit level in this assignment.

```
void updateImportances(WebPage q, Double d);

void changeBaseImportance(WebPage q, Double d);
```

Completing successfully implementations for all the above mentioned methods is sufficient to achieve a Distinction level in this assignment.

```
Vector<WebPage> topTenRanked ();
```

Completing successfully implementations for all the above mentioned methods is sufficient to achieve a High Distinction level in this assignment.

Initialisation

To initialise the WorldWideWeb there is data file called “myTree.txt”. When

setWorldWideWeb(“myTree.txt”) is called the information from myTree.txt is read and a tree of WebPages is created together with a database of base importances. The constructor for doing this assumes a specific format, so if you would like to experiment with different trees please do observe the correct format, otherwise the initialisation is not guaranteed to work.

On each line there is a string, followed by a number, followed by some more strings. Each string and number are separated by a colon. For example:

```
A : 4.4 : D : G : H
```

This means that WebPage with title "A" has base importance 4.4 and refers to pages "D", "G" and "H".

The root node appears on the first place, on the first line;
The root node does not appear anywhere else on any other line;
There are the same number of lines as WebPages;
Each title must appear once at the beginning of some line; and possibly once again in a different line (but not at the beginning).
Except for the root a title cannot appear at the beginning of the line until it has appeared as a reference to some other title.

Notice that in the tests the method setWorldWideWeb(“myTree.txt”) is called before calling the other functions. (Except for testing empty trees.)

What you must hand in

In the submission page on iLearn for this assignment you must include the following:

Submit a single file called WorldWideWeb.java.

Your file must leave unchanged the already implemented functions, and include your implementations of your selection of method stubs outlined above.

Please do not change the contents of the file except to implement the method stubs; do not change the names of the method stubs because the auto-tester assumes the names given. Do not change the package statement.

You may however include additional auxiliary methods in the WorldWideWeb.java if you need them.

Please note that we are unable to check individual submissions and so it is very important to abide by the above submission instructions.

Summary of submission do's and don'ts:
Please make sure that you do not add a package name.
Please be sure to remove all syntax errors before you submit. Occasionally eclipse ignores these errors but the automarker does not.
Please make sure that the methods you implement are all public.
If you decide not to implement a method, please do not remove the method stub.
You must not change the definition of any of the given classes. You may only add additional methods, but you must not add additional data fields to the class definitions or the given method stubs.
Do not submit an eclipse archive, nor a zipped file, nor a .rar file nor any other format than has been stipulated.

Your source code should be submitted via the Assignment 1 link on the COMP2010 iLearn pages.

This assignment will be entirely marked by an automaker. This means that it is crucial that you submit files in the right format and with the right names. Unfortunately we are unable to mark individual assignments, but we will do a trial run of the automaker on **28th April** so that you can check before the deadline that you have the right format. If you would like your programs to be trialled please make sure they are submitted **BEFORE 28th April. (Submissions received ON 28th April may not be collected in time to be trialled.)** You will be able to resubmit as many times as you like before the deadline.

Mark allocation (15%)

This assignment is structured to allow you to decide how much effort you want to expend for the return in marks that you might hope for.

Here is what is required to obtain marks in one of the performance bands for this assignment:

- **Pass** (P, 50%-64%) A successful implementation of numNodes, removeNodes and calculateImportances.
- **Credit** (CR, 65-74%) The P level of implementation plus changeReference.

- **Distinction** (D, 75%-84%) the CR level implementation plus an implementation of updateImportances and changeBaseImportance.
- **High Distinction** (HD, 85%-100%) the D level implementation plus an implementation of topTenRanked.

Late penalty

It is important that you hand your work on in time. Please note that a late penalty of 10% reduction per day late will be applied.