Exploring USGS Earthquake Data Using Binary Search Trees

Kalpathi Subramanian, Dept of Computer Science, The University of North Carolina at Charlotte, krs@uncc.edu

Introduction

The US Geological Survey (USGS) releases data on all earthquakes that occur around the world. These can be viewed directly via maps or as Tweets . For the past few years, our group has been working on developing an infrastructure called BRIDGES, a part of which provides easy to use interfaces to interesting and topical datasets for routine use in freshman/sophomore level computer science courses. In addition, all the basic data structures supported in BRIDGES can be visualized with the data that they store in their structures. Examples of visualizations used in data structures and algorithms courses can be see on the BRIDGES site. Thus, the twin goals of BRIDGES system is to make it easier for sophomore level students to import interesting real-world datasets and view the data and the data structurs they have themselves constructed.

An example BRIDGES Program that creates and visualizes a Binary Search Tree

The example below illustrates a simple BRIDGES program to build a binary search tree representing 50 quake records. To run this program, a BRIDGES Jar file (available at the BRIDGES site) needs to be loaded (the C++ version requires the libCURL library and a set of BRIDGES include files). The visualization will be sent to a weblink output by the program on the console. See the full set of examples at Bridges Tutorials.

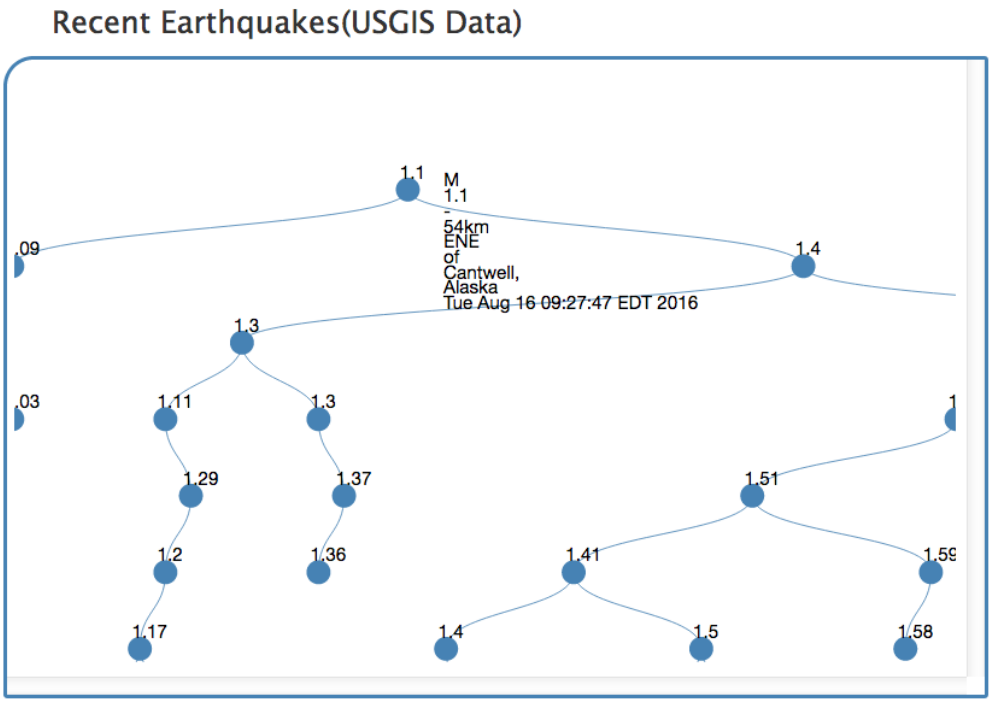
```
import bridges.connect.Bridges;
import bridges.base.BSTElement;
import bridges.data_src_dependent.EarthquakeUSGS;
import bridges.data_src_dependent.Tweet;
import bridges.data_src_dependent.USGSaccount;
import java.util.List;

public class eq {
    public static final int maxElements = 50; //number of tweets

    public static void main(String[] args) throws Exception {

                    // Instantiate a Bridges object - your BRIDGES account
                    // credentials must be used for the 2nd and 3rd arguments
        Bridges bridges = new Bridges (1, "your-api-key", "your-user-id");

                    // set visualization title
        bridges.setTitle("Recent Earthquakes(USGIS Data)");

                    // Set up the earthquake user
        USGSaccount acct_name = new USGSaccount( "earthquake" );

                    // Retrieve a list of (maxElements) earthquake records
        List<EarthquakeUSGS> eqlist = bridges.getEarthquakeUSGSData(name, maxElements );

                    // create a binary search tree object
        BST<Double, EarthquakeUSGS>  bst = new BST<Double, EarthquakeUSGS>

                    // insert the records into the binary search tree
        for ( int i = 0; i < eqlist.size(); i++ ){
            bst.insert(eqlist.get(i).getMagnitude(), eqlist.get(i));
        }

                    // provide BRIDGES a handle to the tree root
        bridges.setDataStructure(bst.getTreeRoot());

                    // visualize the tree and earthquake data
        bridges.visualize();
    }
}
```

where the first call to create the Bridges object requires user credentials (obtained by creating a user account, as described here..

```
Also note that BST class and its methods are to be implemented by the user
and not part of BRIDGES. BRIDGES is designed to provide the building blocks
of the data structure, and the users(students) still implement the full
data structure and the underlying algorithms, as would be done typically
in a course on data structures/algorithms.
```

Quake Data Visualization

```
If the program runs successfully, then a tree visualization will be displayed
in the link that is provided to the user, with the key values displayed
beside each node. Mouseover of any node will display the contents of its
label (this is defined by the user, via the Element::setLabel() method). An
example visualization is shown below. See a live example on the BRIDGES website.
```



Project Description

The project will explore the use of binary search trees and their underlying algorithms using the USGS earthquake dataset and the BRIDGES infrastructure. We have set up an asynchronous process to collect and parse the USGS earthquake tweets; the data gets stored in a MongoDB and is transparent to the user. *Thus, each time a driver is run to extract the quake data, by default the latest set of quakes is extracted, enabling a user to see the most recent quakes!* To access the data, the user issues the following calls from his driver program (after creating the BRIDGES Object:

```
USGSaccount acct = new USGSaccount( "earthquake" );
List eqlist = bridges.getEarthquakeUSGSData(acct, maxElements);
```

These calls retrieve maxElements (user specified) earthquake records from the DB and are returned in a Java list of objects of type EarthquakeUSGS (which is part of BRIDGES). At this point the user is free to use the data as part of his project. One additional call is required to store the object as part of the BSTElement (which is subclassed from Element), for instance,

```
root.setValue(eqlist[k]);
```

Example Project Tasks

I have used this data several times in my data structures and algorithms courses. The project was broken up into 3 parts:

1. Implement the binary search tree ADT using the text book's implementation (carried very little credit since it was already in

the text book).
2. Make the needed changes to use the BSTElement class for the nodes and to visualize a set number of earthquakes (using the driver described above). Earthquake magnitude was used as the search key for the tree. Label the nodes to illustrate (mouseover operation illustrates the label) the location, date, time of the quake. See the earthquake example posted on the BRIDGES website
3. The last part of the project included the following tasks:
    a. Find and highlight the largest earthquake
    b. Find and highlight the smallest earthquake
    c. Find and highlight the quakes within a user specified range.
    d. Find and highlight the quakes that satisfy a date (by month, by year, etc)
    e. Find and highlight quakes by a specified location(for instance, Alaska).

Intended Students

The project will clearly be appropriate for a course on data structures (we have received very positive feedback via surveys), but the above tasks can be simplified for lower level students (CS1, CS2) by providing an implementation of the ADT and critical functions and engage the students with the visualization(CS0/CS1 - might just run the application with an implemented interface to just look at the current quakes, while focusing on performance/efficiency issues and the motivation behind search trees. More advanced students can use the AVL Tree Element object (part of BRIDGES) to build balanced binary search trees, perform tree rotations(this was a project in our algorithms course), etc. The visualizations help engage students and also serve as a more attractive way to check their work. Grading is also via interactive demos, promoting more direct interaction with students.

Provided Materials/Documentation

1. The BRIDGES website has complete documentation (C++ and Java implementations are supported); this is an ongoing project so full support to instructors and students are provided by the BRIDGES development team for issues relating to BRIDGES setup.
2. Documentation on getting started with BRIDGES with various IDEs are provided(not a complete list yet).
3. BRIDGES is currently hosted by an external server(Heroku and soon on Amazon), thus reliability should be high in terms of server performance.
4. **Sample Program** (as listed above)
    a. README - describes how to compile run the sample program, generate visualization (snapshot above)
    b. Source files (Dictionary.java, BST.java, eq.java)
    c. BRIDGES Jar file (bridges-2.2.0.jar)

Meta-Data

Summary	Project enables working with a real-world dataset - earthquake data from USGS and applying it to binary search trees, generating visualizations of the tree and the data of the latest quakes that occured **minutes** ago.
Topics	binary search trees or AVL trees, algorithms on binary search trees, generic programming
Audience	Appropriate for CS2 or Data Structures courses, but also perhaps CS1 (with some components implemented and provided to students).
Difficulty	Intermediate to advanced assignment, 1-2 weeks, depending on the parts of the assignment assigned by the instructor.
Strengths	Easily work with a real-world scientific data set, without getting bogged down in parsing or fighting with I/O; rather focus on core algorithms on search trees, and quickly produce visualizations of the data structure in a way that is easily shared(via weblinks) on any device; BRIDGES system is well documented with live examples.
Weaknesses	Some effort to setup BRIDGES - creating account, integrating a Jar file into IDE (sample program is provided to make this easier); if using C++ version, then the user must build the Curl library for the visualization component. Fully supported by the BRIDGES development team to ease the learning curve.
Dependencies	Need to be reasonably competent in Generic Java or C++(Bridges uses templates); however, this is compatible with most textbooks on data structures(BRIDGES design roughly follows Cliff Shaffer's Data Structures textbook .
Variants	An advanced version using AVL trees can be used for courses on Algorithms, by maintaining a balanced search tree, exploring tree rotations, extracting subsets of the tree, tree transformations. For CS1, these can be used to demonstrate and explain issues of performance.