Rivers Are Trees Design Document

By: Damon L, Stuart G, Jack O, Geoff S

Initial Design

Binary Tree Design

A standard binary tree was used for the project. Because the data in the tree is not easily ordered in the way a Binary Search Tree is, we picked a different convention to use when inserting elements, where the continuation of an existing river or a dam or lake in the way of the river, would be placed in the left child of a node. A tributary or other split from the river would become the right child.

We do not use one Node class because each node might contain very different data. Instead, we have an abstract class for the Node, with three subclasses for rivers, dams, or lakes. Each subclass contains its data and methods to print or serialize it.

Showing the Tree

We decided to use a level-order traversal when printing the entirety of the tree to the screen. It was chosen because it separates different levels of the tree and can better show the various splits in the river while still being easy to implement and test.

Navigating the Tree

We re-used some existing code from Project 1 to offer the user a list of options. When navigating the tree, the base options would be to go up to the parent, go to the left child, go to the right child, go back, or quit the program. Because parents and children may not always exist, we check each edge case and remove the relevant option if needed.

Since each node doesn't keep track of its parent, we needed some external way to get the parent of a node. Because of our ordering scheme, a traditional search was not possible. We used a stack to create a history of what nodes were visited. We push or pop from history when going up or down a level. The stack also made checking edge cases and finding the current node convenient.

Changes to Design

We decided to have certain specifications regarding which rivers and dams we wanted to include. The rivers were included if they were not over 100 km in length. For the dams, if they produced less than 500 megawatts in power or were under 20 m in height, they were not included. Lakes were not included if they had a surface area of less than 100 km^2. If data was missing from Wikipedia, we did not include the feature in our data. We included 18 rivers, 12 lakes, and 23 dams in total.

We initially planned to read the binary tree data from a file. However, this posed several complications, including difficulty writing that data in a human-readable way. We decided to manually build the binary tree in the Tree class's constructor, which, while not ideal, is simple for us to program.

Due to time constraints and numerous challenges in finding the proper placement of dams, tributaries, and lakes, we ended up only including tributaries in the tree. The rest have data but are not inserted. There is no indication of how far away each tributary is from each other because finding data on the start of each proved to be very difficult.

How to Use the Program

Compiling and Running

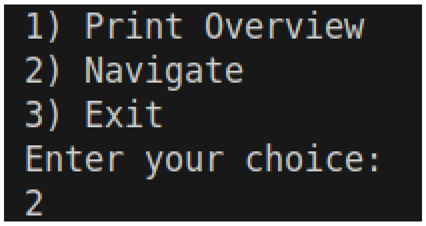
We have included a Makefile in our program to make it easier to run (on Linux).

To compile and run the base program: make

To compile and run the unit tests: make test

To clean up any executable files: make clean

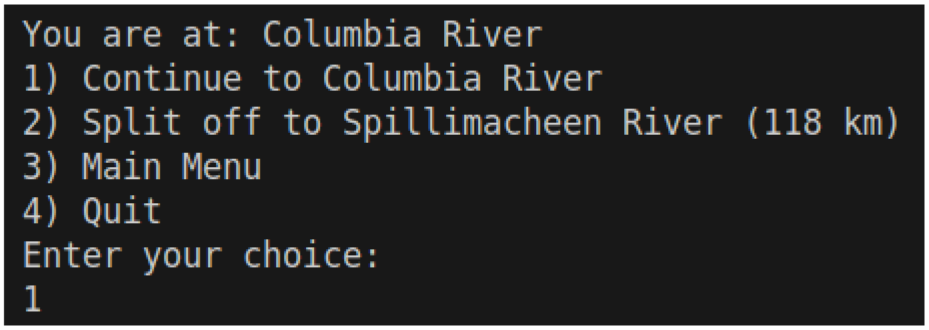
Main Menu



```
1) Print Overview
2) Navigate
3) Exit
Enter your choice:
2
```

Our menus use numbers for choices. For the main menu, 1 will print a representation of the entire tree. 2 will allow you to navigate through the tree, and 3 will exit the program.

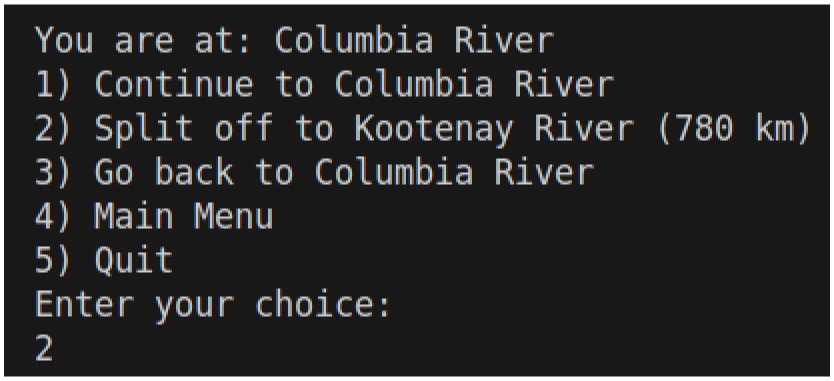
Navigating the Tree



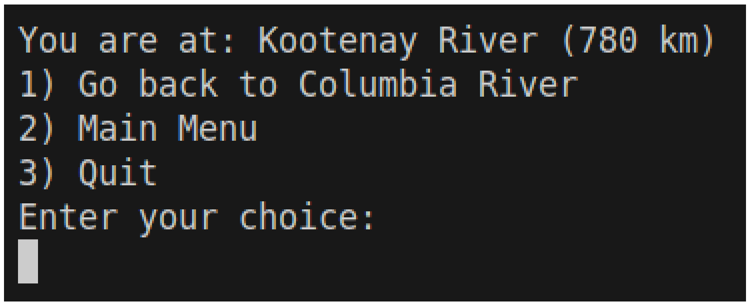
```
You are at: Columbia River
1) Continue to Columbia River
2) Split off to Spillimacheen River (118 km)
3) Main Menu
4) Quit
Enter your choice:
1
```

Our navigation will start at the root node. You will see the node you are currently on at the top.

You will see options to continue the Columbia, split off to a tributary, or go back to where you came from. These options will only appear if you can use them, so be careful what you type: options will move around as you navigate through the tree.



```
You are at: Columbia River
1) Continue to Columbia River
2) Split off to Kootenay River (780 km)
3) Go back to Columbia River
4) Main Menu
5) Quit
Enter your choice:
2
```



```
You are at: Kootenay River (780 km)
1) Go back to Columbia River
2) Main Menu
3) Quit
Enter your choice:
```

At each point in the navigation, you can either go back to the main menu or quit the program.

Coding Standards

We followed the CS department style guide linked in the course Canvas page.