TrawlExpert: Tool for Watershed Biological Research

Trawlstars Inc. (Group 11)
Lab section: L01
Version: 1.0
SFWRENG 2XB3

Christopher W. Schankula, 400026650, schankuc
Haley Glavina, 001412343, glavinhc
Winnie Liang, 400074498, liangw15
Ray Liu, 400055250, liuc40
Lawrence Chung, 400014482, chungl1

April 9, 2018

Revision History

Revision	**Date**	**Author(s)**	**Description**
1.0	05.04.18	HG	created
1.1	08.04.18	HG	algorithmic analysis added

By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals in the group is their exclusive work as a group and we consent to make available the application being developed through SE-2XB3 project, the reports, presentations, and assignments (not including my name and student number) for future teaching purposes.

Team Contributions

The individual contributions of each team member are described below. Subteam B indicates an algorithmic focus in a member's efforts while Subteam A indicates a focus on data parsing and user interface development. Although the contributions have been separated such that each task is recorded under one contributor, members often overlapped duties and designed modules together.

Name	**Role**	**Contributions**
Lawrence Chung	Head of Room Booking Subteam B Member	Implemented the depth first search and connected components algorithms.
Haley Glavina	Meeting Minutes Administrator Subteam B Member	Implemented the red-black tree, quickselect, and mergesort algorithms. Designed the final presentation powerpoint, recorded and submitted all meeting minutes, and assembled the final design specification in LaTeX.
Winnie Liang	Project Log Administrator Subteam A Member	Implemented the module responsible for parsing out data to create related objects, implemented taxonNode ADT. Led user interface development, set up tomcat files and directory structure, handled communication between the Google Maps APIs and javascript code. Overlooked project log entries.
Ray Liu	TA & Professor Liaison Subteam A Member	Implemented Record ADT, Date ADT, parsing API calls for WORM API, RangeHelper for Basic Search, Histogram.
Christopher Schankula	Team Leader Subteam A Member	Determined the goals for each meeting, implemented the k-d tree algorithm, and handled server communication.

Abstract

TrawlExpert is a powerful tool to enable researchers to analyze and filter large datasets from fish trawl surveys in order to perform environmental research on fish and invertebrate populations. The tool gives researchers the ability to intelligently filter and query datasets based on biological classification such as family, genus or species, or based on location or timeframe. Advanced outputs display data as a histogram or geographical map, each depending on population abundance as a function of time and spatial distribution. Additionally, *TrawlExpert* provides a tool for finding local subpopulations within a larger query. A dataset of thousands of Great Lakes trawl surveys from 1958-2016 will be used as a demonstration of *TrawlExpert*'s capability to help researchers narrow down large datasets and glean data which pertains to their research. *TrawlExpert* will be designed to be used easily and effectively as the first step in a groundbreaking climate and ecological research pipeline.

Contents

1 Project Scope

1.1 Objective

Provide a statistical and visual tool for the analysis of water ecosystems, based on scientific water trawl data. Gives researchers with tools to analyze large datasets to find patterns in fish populations, including the plotting of historical population data on a map, the analysis of population trends over time and the determination of subpopulations of a certain biological classification.

1.2 Motivation

The diminishing of fish populations in the Great Lakes became a problem in the latter half of the 20th century, with the total prey fish biomass declining in Lakes Superior, Michigan, Huron and Ontario between 1978 and 2015 (Kinnunen, 2017). Annual bottom trawl surveys involve using specialized equipment to sweep an area and are used to determine the relative temporal variation in stock size, mortality and birth rates of different fish species (Walsh, 1997). These surveys are performed annually and often have hundreds of thousands of records, making manual analysis infeasible. The ongoing protection and development of the Great Lakes water basins is considered an important topic for scientists in both Canada and the United States, as evidenced by grants such as the *Michigan Sea Grant* (Michigan State University, 2018).

TrawlExpert will give researchers tools to filter through these large amounts of data by allowing them to search through data based on class, order, genus, family or species. This will help support scientific researchers and fishing companies as they study fish populations. These studies help inform initiatives to preserve fish populations and conduct their business in an environmentally friendly way going forward. As more data is collected on an annual basis, TrawlExpert can easily be injected with the new data and will adjust and scale accordingly, combining the new data with the old data for continued analysis.

TrawlExpert will also analyze the trawl data to find connected subpopulations within the data, giving researchers tools to analyze the portions of the water body that contain different populations and even track these specific subpopulations over time.

The focus of the project will be to develop these unique data searching and querying tools as a first step in a complete trawl survey analysis. For a complete analysis, tools like stratified statistical analysis are required by the researcher (Walsh, 1997). For purposes of maintaining a manageable scope for this project, the implementation of advanced trawl survey scientific and statistical analysis tools will be relegated to future developments.

1.3 Dataset

The test dataset that will be used for purposes of this project is the *USGS Great Lakes Science Center Research Vessel Catch Information System Trawl* published by the United States Geological Survey (United States Geological Survey, 2018). Compiled on yearly operations taking place from early spring to late fall from 1958 until 2016, the dataset contains over 283,000 trawl survey records in the five Great Lakes, including the latitude and longitude co-ordinates and biological classification such as family, genus and species.

2 Implementation

2.1 Classes and Modules

The implementation involved over 30 classes implemented in Java. Additional JavaScript and html files were used to create a sophisticated user interface. For a description of each class and module used, Java documentation can be viewed at

2.2 Class Organization

The following UML diagrams depict the organization and use-relations of all classes in the program. Two UML state machine diagrams are included to describe the states and transitions within the *BioTree.java* and *Main.java* class. Since the *Main.java* class is a console version of the final server implementation, the states shown in its UML state machine diagram are analogous to the states of the final *TrawlExpert* product.

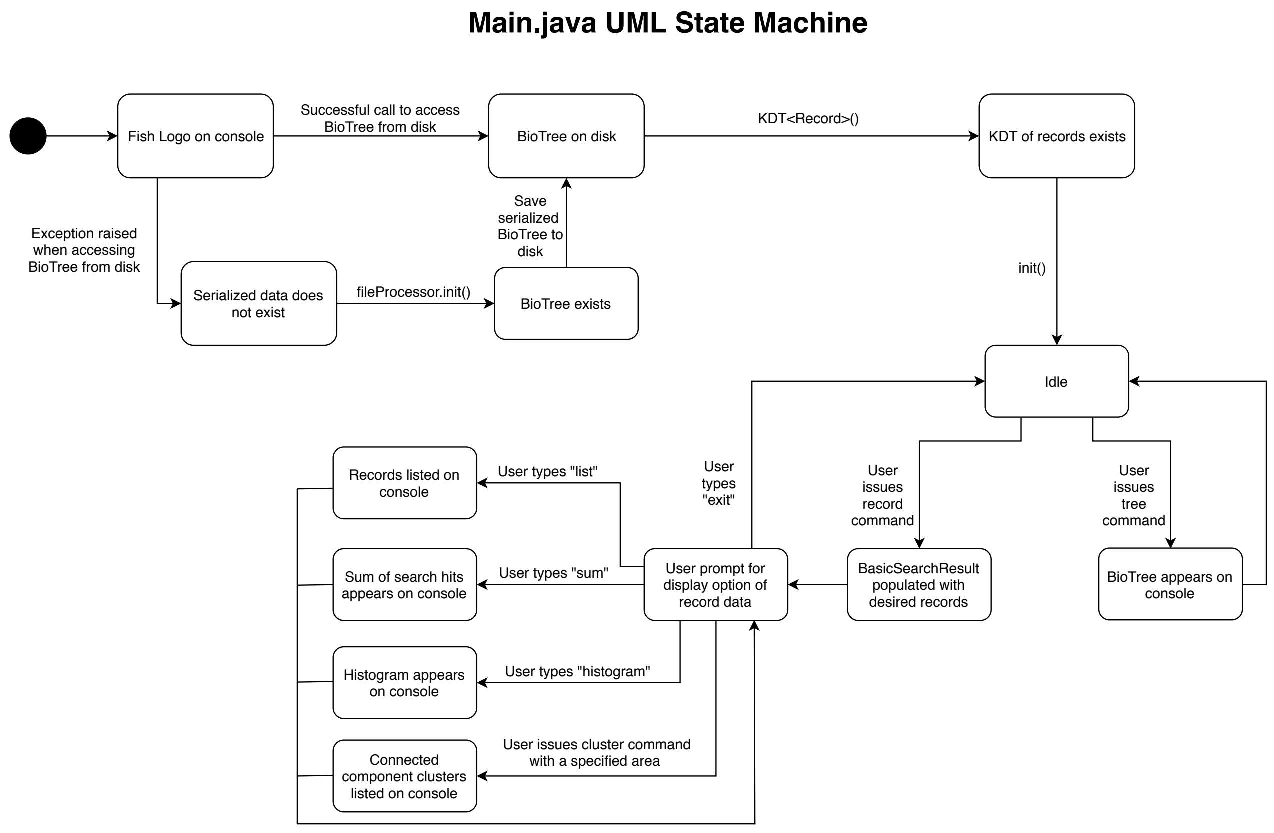
Main.java UML State Machine
Fish Logo on console
Successful call to access BioTree from disk
BioTree on disk
KDT<Record>()
KDT of records exists
Exception raised when accessing BioTree from disk
Save serialized BioTree to disk
Serialized data does not exist
fileProcessor.init()
BioTree exists
init()
Idle
User types "exit"
User issues record command
User issues tree command
Records listed on console
User types "list"
Sum of search hits appears on console
User types "sum"
User prompt for display option of record data
BasicSearchResult populated with desired records
BioTree appears on console
Histogram appears on console
User types "histogram"
Connected component clusters listed on console
User issues cluster command with a specified area

Figure 1: UML State machine diagram for *Main.java*, a class that provides console access to the *TrawlExpert*'s main functions. This class accepts search criteria from a user to produce a list of search results, depict a histogram of the records in that result, and compute a count of the search hits.

2.3 Maintaining Generality

A common theme among *TrawlExpert* classes is the use of lambda functions. Lambda functions provide the capacity for parameterized object comparison or parameterized value access. This maintains the generality, and therefore reusability, of each class by allowing for generic types in class definitions. Type(s) of the input(s) and the how input object(s) are used only become assigned when the function is used.

2.3.1 General Compare

The *GeneralCompare* interface can be found at */sort/GeneralCompare.java*. This interface includes a *compare* function that takes two generically typed inputs and produces an integer output. When *GeneralCompare* is used in other classes, a compare function (the lambda function) is used to instantiate the expected input type and designate how the integer result must be calculated. This allows reuse of the interface among modules that perform comparisons of differently typed objects. Two records consisting of a fish species, date of observation, and geographic location can be compared based on lexicographic order of their names, date, or proximity to some location. *GeneralCompare* enables the comparison of record objects based on any of these parameters.

2.3.2 Field

The *Field* interface can be found at */search/Field.java*. This interface includes a *field* function that retrieves a key (a generic type) from a generically typed input object. Similar to *GeneralCompare*'s *compare* function, *field* is a lambda function. The field interface is used to perform searches in a tree of records that have been sorted by variable attributes from each record. The lambda function specifies which attribute to access when searching through the tree.

2.3.3 General Range

The *GeneralRange* interface can be found at */sort/GeneralRange.java*. This interface includes a *isInBounds* function returns an integer to describe if a record is member to a subset of the search results. The input has a generic type, rather than *Record* type, to satisfy reusability. The lambda function uses the range itself to perform conditional checks about whether the input object is below, within, or above the range. A return value of -1 indicates it is below, 0 indicates it is within, and 1 indicates it is above the range.

3 Algorithmic Opportunities

The *TrawlExpert* was made possible by the use of vaious algorithms studied in *SFWRENG 2C03: Algorithms* offered at McMaster University. These algorithms include *Red-Black Tree* for searching and *Merge Sort* for sorting objects. Additional algorithms outside of the course scope were implemented to optimize the program, they are described below.

3.1 Quick Select Algorithm

A modified form of the *Quick Sort* algorithm that returns the k^{th} largest element of an unsorted array. Similar to *Quick Sort*, *Quick Select* randomly chooses a partitioning element to sort the array such that all elements smaller than the partition are left of it, and larger elements are to the right. However, rather than recursively sorting both halves of the partitioned array, *Quick Select* only sorts the half containing the k^{th} index. The algorithm terminates once the partitioning element ends up at the k^{th} index of the array, the value of this element is returned.

This algorithm is implemented in */sort/QuickSelect.java*. It is used during the construction of *k-d tree*s which require the frequent division of an array into two equally sized halves. By finding the median element of an array, it is partially sorted into equally sized small and large halves. The *Quick Select* class implements a *median* method to simplify its usage in *k-d tree* construction.

Using *Quick Select* rather than *Merge Sort* has optimized the *TrawlExpert*. When using *Merge Sort* during *k-d tree* construction, an array must be fully sorted before retrieving the median element. *Quick Select* only partially sorts the array before reaching the median and has reduced *k-d tree* construction from 40.083 s to 0.56 s

3.2 K-d Tree Algorithm

3.3 Graph Algorithms

Graph algorithms were used to support advanced searching features. Firstly, the biological classification of each organism forms a tree from which species in the same genus, for example, can be located.

Secondly, a graph algorithm was used to find connected components among search results. Nodes are connected together based on their distance to surrounding points (Tom10, 2012). Depth-first search was used to determine connected components (Broder et al., 2000).

References

Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2000). Graph structure in the web. *Computer networks*, 33(1-6):309–320.

Kinnunen, R. (2017). Great Lakes prey fish populations declining. http://msue.anr.msu.edu/news/great_lakes_prey_fish_populations_declining_msg17_kinnunen17.

Michigan State University, U. (2018). Michigan sea grant. http://www.miseagrant.umich.edu/.

Tom10 (2012). 2d point clustering. https://stackoverflow.com/questions/3937663/2d-point-clustering.

United States Geological Survey (2018). USGS Great Lakes Science Center Research Vessel Catch Information System Trawl. https://www1.usgs.gov/obis-usa/ipt/resource?r=usgs_glsc_rvcat_trawl.

Walsh, S. J. (1997). Efficiency of bottom sampling trawls in deriving survey abundance indices. *Oceanographic Literature Review*, 7(44):748.