**Final Project Report for Even Semester 2020**

*Submitted as course project of*

**DATA STRUCTURE AND ALGORITHMS**

**COMP6571**



*Written by:*

**Karel Bondan Andoro Herdito – 2440032373  
Leon Jayakusuma – 2440032392  
Nathanael Jason Limantara - 2440047476**

*Under the Guidance of:*

**MARIA SERAPHINA ASTRIANI, S.Kom., M.T.I.**

**Lecturer of Data Structure and Algorithms**

**School of Computer Science**

**Binus International University**

1. **Project Background**

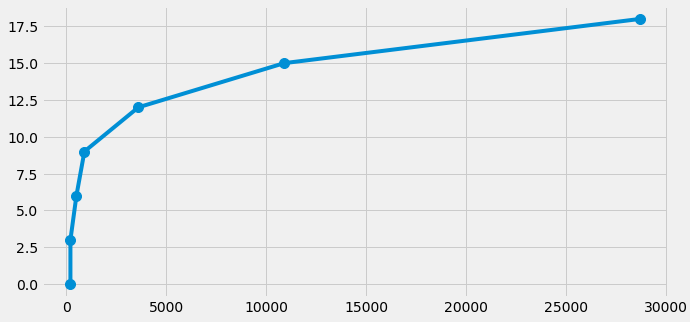
Binary search tree is a difficult subject: its concepts are hard to learn especially its algorithms. It can be very confusing especially when it has become a complex tree, which means that it has a lot of nodes. Another thing that is hard to learn is the tree’s algorithms: insert, delete, search, and traversals. One might have to draw it on a sheet of paper and try to visualize them using their understanding, which can mislead themselves if they got the wrong understanding of the concepts. For this problem, we came up with an idea: to make an application that can visualize binary search tree’s algorithms and the tree itself. We understand that this is a difficult subject to learn, so we want to ease people who are currently learning about the subject so they can understand the concepts of binary search tree easier. The features of this application include auto generation of a binary search tree with size *n,* insertion, deletion, search, breadth-first traversal, pre-order traversal, in-order traversal, and post-order traversal.

1. **Proposed Data Structure**

Our project is an application that can visualize binary search tree visualizer and its algorithms, which means that we must use binary search tree data structure to bring this application into realization. If we were to use a tree data structure or a binary tree data structure, then the whole concept of the project will fail because what we are trying to achieve are not present in those two data structure. Other than that, if binary search tree’s algorithms are to be used in other data structure other than binary search tree data structure itself, then they will all fail because they operate on a structure that is not made for them. Binary search tree algorithms only works on the binary search tree data structure itself. Therefore, binary search tree data structure is the one that we proposed for this project, since it is the most suitable and natural data structure to be used in this project.

1. **Theoretical Analysis of the Data Structure**
2. **Time Complexity for Insertion, Deletion, and Search Operations**

We ran a few insertion test with binary search tree of size 2, 4, 8, 16, 32, 64, and 128. The time taken for each insertion is 200, 200, 500, 900, 3600, 10900, and 28700 nanoseconds respectively. To ease us in visualizing the time complexity, we have made a graph to represent each of the resulted time:



As we can see, the graph decelerates as the size of the binary search tree increases. This means that the time needed to insert the nodes is O(log n) because insertion in binary tree uses the divide and conquer method, thus making the processing time linear as the size goes exponentially. When the current value to be inserted is less than the current node in the tree, then it will go to the left side of the current node, completely ignoring the right side/right sub-tree of the tree. The same happens when the value to be inserted is bigger than the current value in the tree: it will go to the right side of the tree, completely ignoring the other half of the tree which is the left sub-tree. We only show one example, which is the insertion of the tree, because the other two algorithms –deletion and search **–** apply the same method. Since they used the same method, they will produce the same time complexity which is O(log n).

1. **Binary Search Tree’s Traversal Algorithms**

These are the execution time comparison shown in nanoseconds for each traversal algorithm with a binary search tree of size 10, 100, 1.000, and 10.000:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Size | Breadth-First | Pre-Order | In-Order | Post-Order |
| 10 | 5300 | 2900 | 2400 | 5800 |
| 100 | 17500 | 10000 | 9800 | 9800 |
| 1.000 | 130200 | 92200 | 86200 | 100800 |
| 10.000 | 1.1277e+06 | 1.0984e+06 | 843,400 | 920,200 |

Here, we can see that breadth-first traversal took the most time compared to the other algorithms except for the first one, which is the binary search tree of size 10. This is because breadth-first traversal uses another data structure as its base, which is the queue data structure. It has to do the push and pop operation of the queue repeatedly until the queue is empty, therefore taking twice the time to process the data that has been pushed inside the queue.

1. **Program Manual**
2. **Introduction**

This project uses QT Creator as its foundation. If you have not install QT Creator on your PC and want to know the workflow of the application by using the debug feature, please install it first. Otherwise, you can directly launch the application by going into the “Release” folder and double clicking “BST\_Visualizer.exe.” Information regarding the installation of QT Creator will be discussed in the 2nd section of this manual. The Binary Search Tree Visualizer application uses a depedency for its visualization called Graphviz. Graphviz is a library that can be used to make graphs, including but not limited to binary search tree. Since this project is made using Windows, this manual is and will mostly be addressed to Windows users. Note that this application has not been tested yet in other operating system other than Windows, thus it may or may not work outside the Windows operating system.

1. **Installing Graphviz**

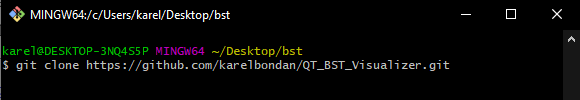
To install this depedency, download the installer package from <https://graphviz.org/download/>. Go to the section according to your operating system, and download the necessary files from there. Open the downloaded package and install the application. **For users that want to directly open the application without QT Creator (by launching the application via “Release” folder), they must install GraphViz to C:\Program Files\Graphviz, otherwise the application will not work**. However, if you want to install it to a different location, please open this application via QT Creator. You can skip the 6th section of this program manual if you decide to install Graphviz on the suggested location. If you decide to install it on another location, please read and do not skip the 6th section of this program manual.

1. **Installing QT Creator**

QT Creator is a powerful GUI application development that uses C++ for its backend code. To install this application, please visit <https://www.qt.io/download-open-source>. Scroll down until you are at the bottom of the page and press the download button. Wait until the download has finished, and run the installer to install the application on your PC. Since this is a community edition version of the application, you are required to make an account, so it is best for you to prepare an email to be used as the registration process. Follow the account registration and installation process in the package installer until it is finished. The installation tutorial video can be accessed from the link in its respective section.

1. **Cloning the Repository**

This project is hosted on GitHub. Therefore, you can clone the repository and place it somewhere on your PC. Open GitBash inside the directory where you want to clone the repository, then type git clone followed by <https://github.com/karelbondan/QT_BST_Visualizer.git> with a single space into GitBash and execute it by pressing Enter on your keyboard.



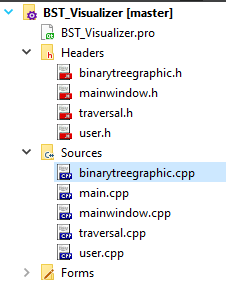
Wait until the process is finished.

1. **Opening the Project**

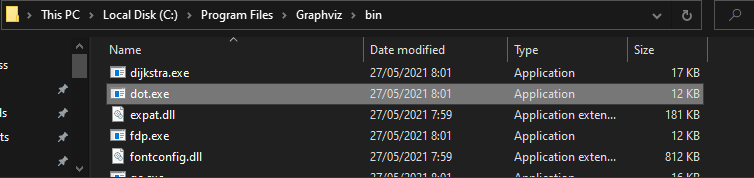
To open the project, open QT Creator that has been installed and press the “Open” button .. If the button does not show up, please press the “Welcome” button on the left bar of the application . Navigate to the directory where you have cloned the repository. After you have found it, open a folder named “BST\_Visualizer” and double click the “BST\_Visualizer.pro” file. If you installed QT correctly, it should automatically select a compiler kit for the project. After you are done configuring, press the “configure project” button and wait until it has finished loading up the project. Feel free to open and look at any source file that are available inside the project. To better understand on how to open and configure the project, please watch the tutorial video that can be accessed from the link that has been attached in its respective section in this report.

1. **Changing the Graphviz Depedency Location**

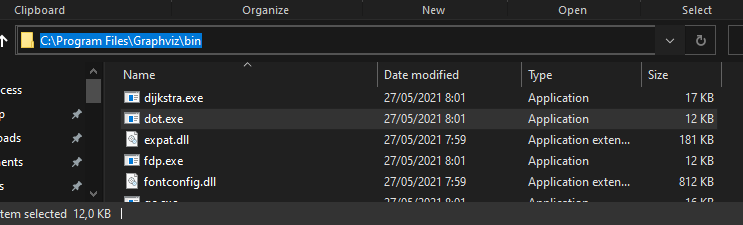
If you previously decided to change the install directory of Graphviz, please open “binarytreegraphic.cpp” file by expanding the project folder. You can click the arrow pointing right beside it to expand the folder.



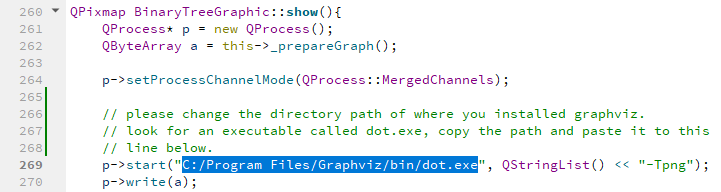
Double click the file and scroll down until it reaches the bottommost part. There, you can find a commented section telling to change the path location. Open the directory where you installed Graphviz before, and locate the executable file called “dot.exe” inside the “bin” folder.



Copy the path by clicking on the address bar and pressing CTRL+C on your keyboard, or simply right click on the address bar and press “copy”.

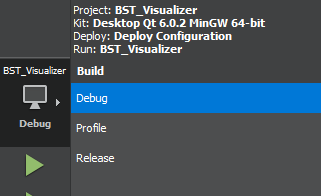


Paste it into the corresponding line inside “binarytreegraphic.cpp” file, make sure to change the backslash (\) into a normal slash (/) (C:\Program Files\Graphviz\bin\dot.exe to C:/Program Files/Graphviz/bin/dot.exe):

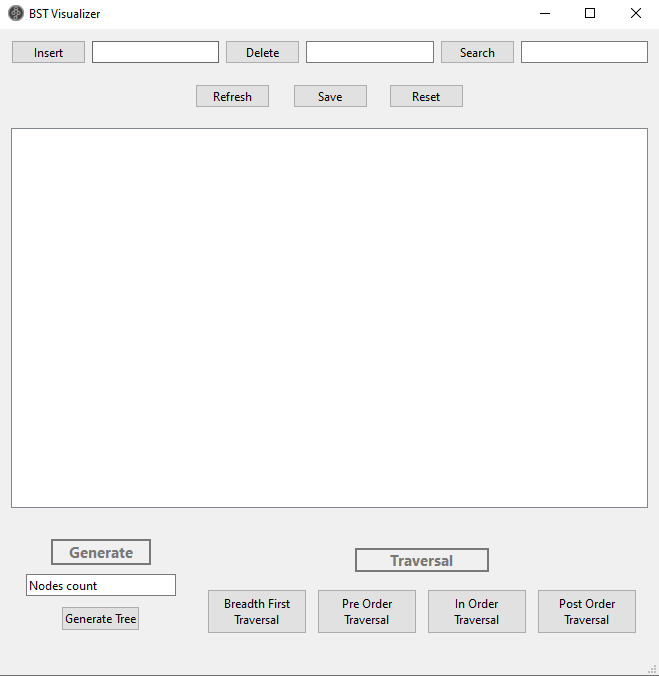


1. **Running the Application**

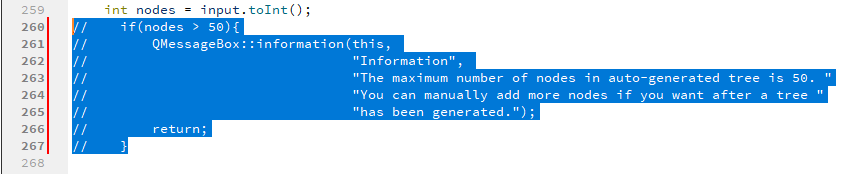
If you want to run the application, make sure to first set up the run mode to “debug”. This option can be accssed on the bottom left of the application, above the “play” button.



After you have selected the run mode to “debug”. Press the “play” button beneath it to run the application. Give it some time. If everything is done correctly, a new application window will open titled “BST Visualizer.” As stated before, other than opening the application via QT Creator, you can also open the executable file from the “Release” folder, which will open the exact same application:

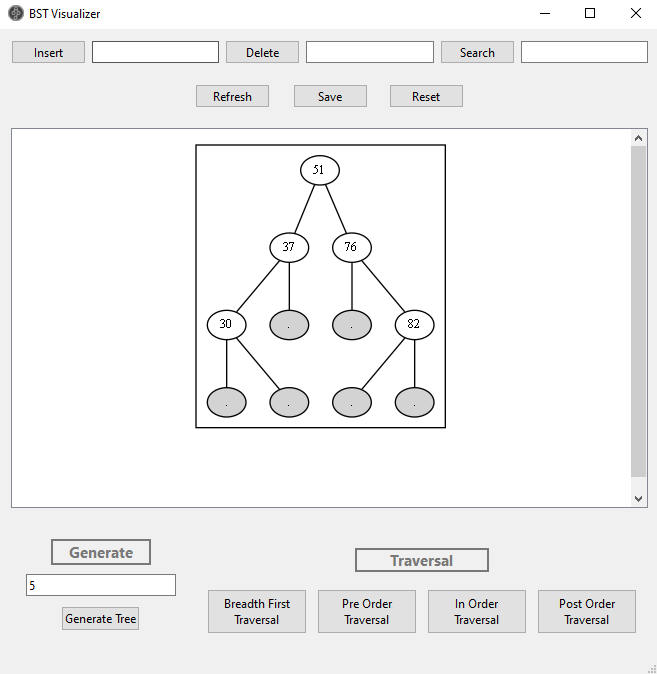


Feel free to do whatever you like with it. A limitation to note is that the auto-generate feature of this application can only accept up to only 50 nodes. This is a failsafe method to avoid a bug where the tree will not show because of the limited processing time (500 milliseconds). As the tree gets bigger, the processing time will take longer and thus it can create a bug where Graphviz hasn’t finished processing the tree but it already needs to show the tree to the user. You can add more nodes after the auto-generation however, but keep in mind that the bug might happen. If you want to remove the 50 nodes limitation, open “mainwindow.cpp” and comment out these lines:

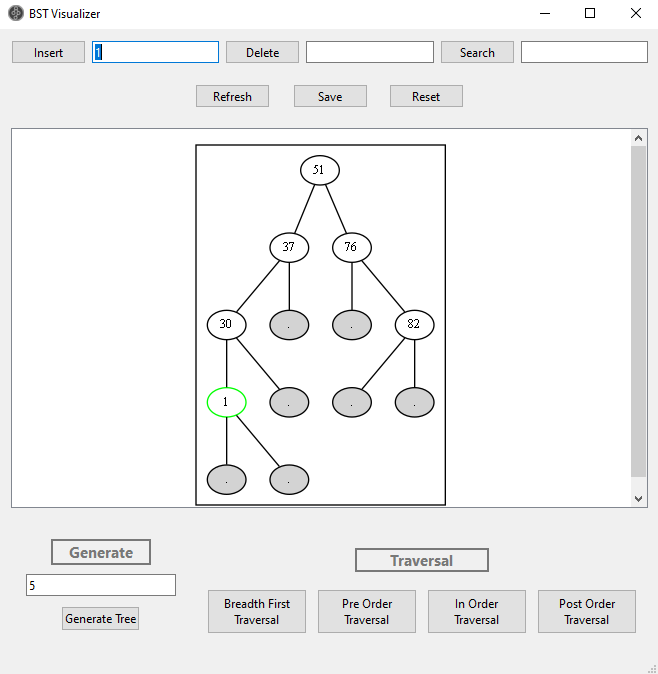


The application’s random number generator is limited to 100 with unique values, thus the maximum number of nodes that can be generated with unique values are 100. **Please do not change the randomizer as it can really break the application**. The link to the application demo is stated in its respective section in this report.

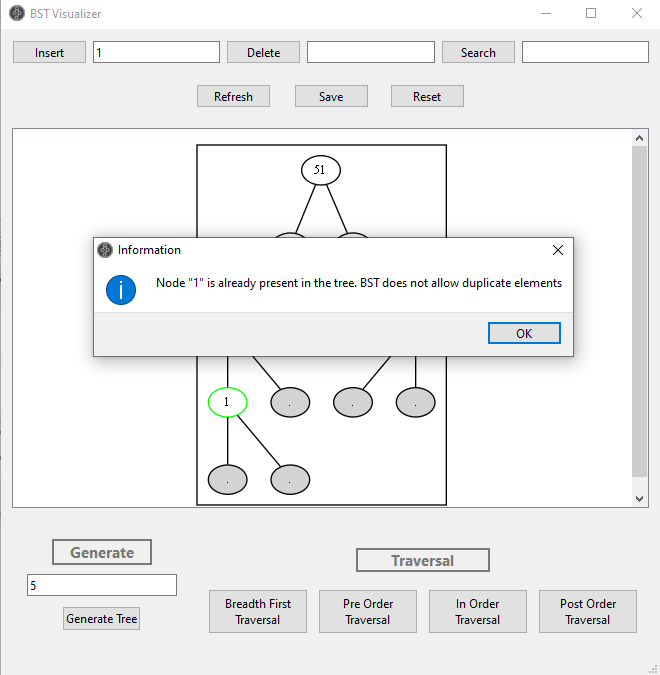
1. **Results of Application Executions**
2. Result of execution: Auto-generating a new binary search tree with size 5 (5 nodes).



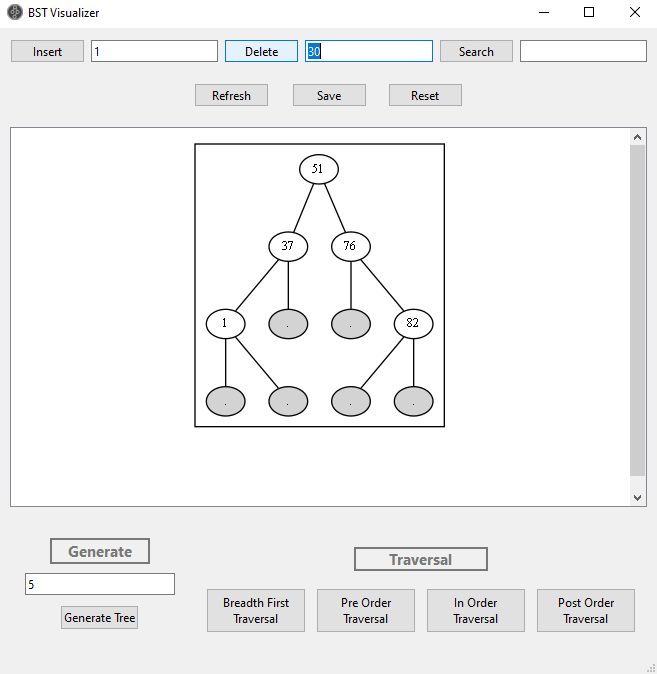
1. Result of execution: Inserting a new node



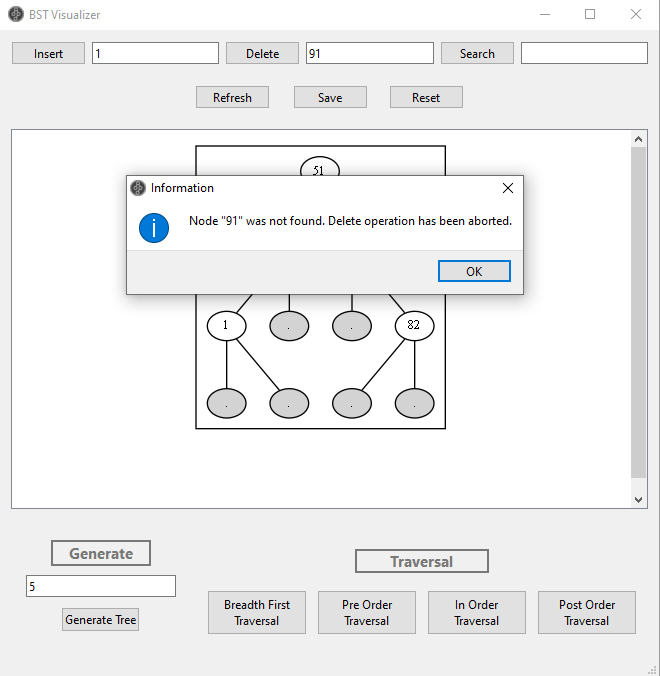
1. Result of execution: Inserting a new node that has already existed in the tree



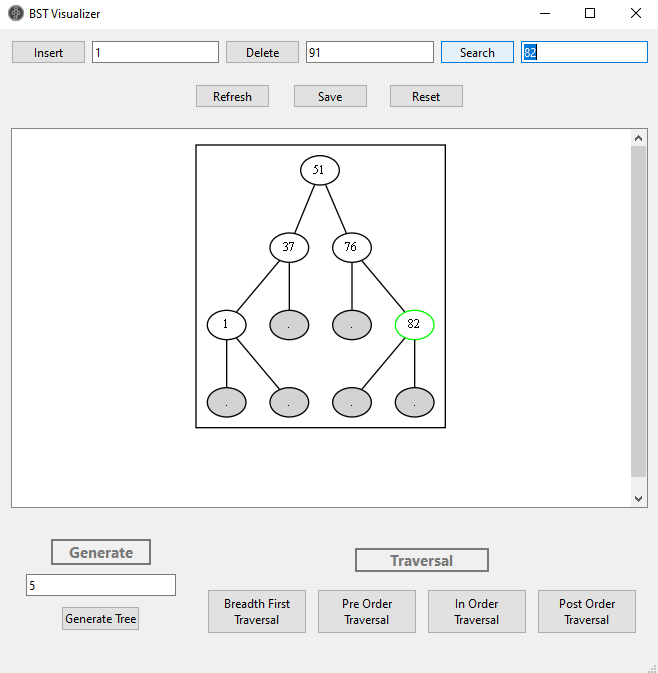
1. Result of execution: Deleting a node



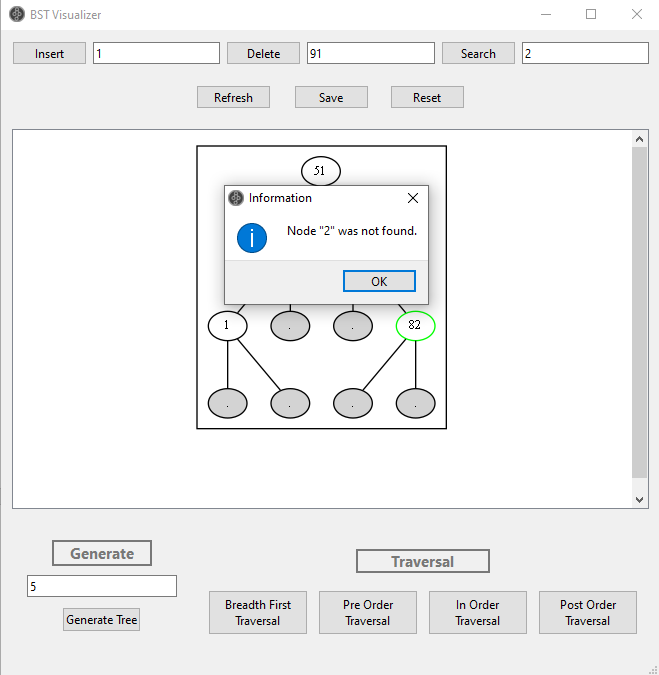
1. Result of execution: Deleting a node that does not exist



1. Result of execution: Searching a node

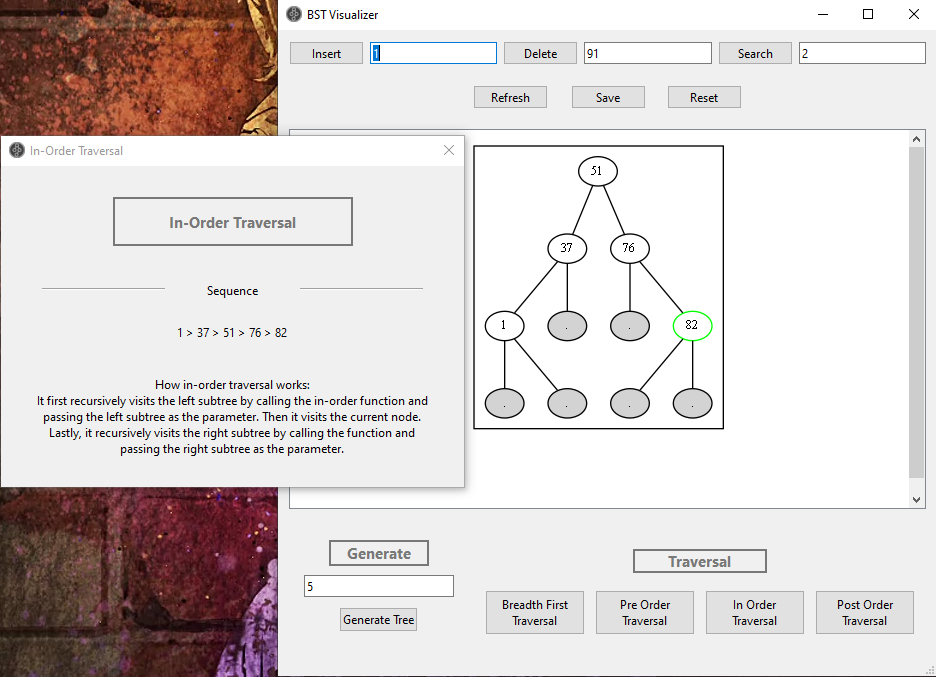


1. Result of execution: Searching a node that does not exist



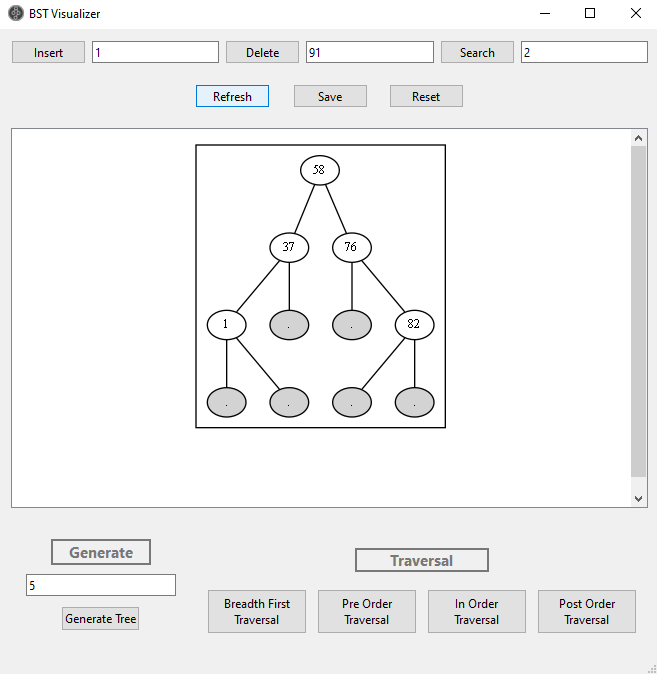
1. Result of execution: Traversals

It will open a new window with the traversal information that the user chose. In this screenshot below, the in-order traversal was selected. A new window with the in-order traversal information such as the sequence and how the traversal works will be shown as in the screenshot below:

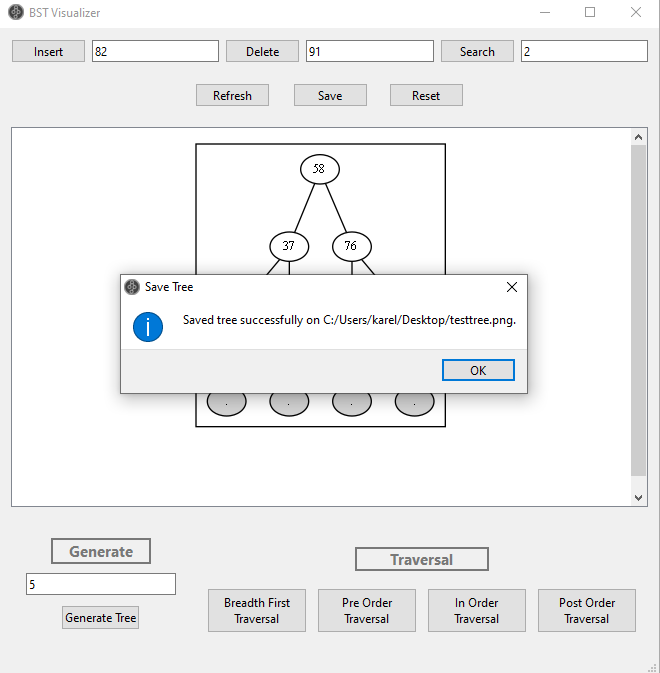


1. Result of execution: Refreshing the tree

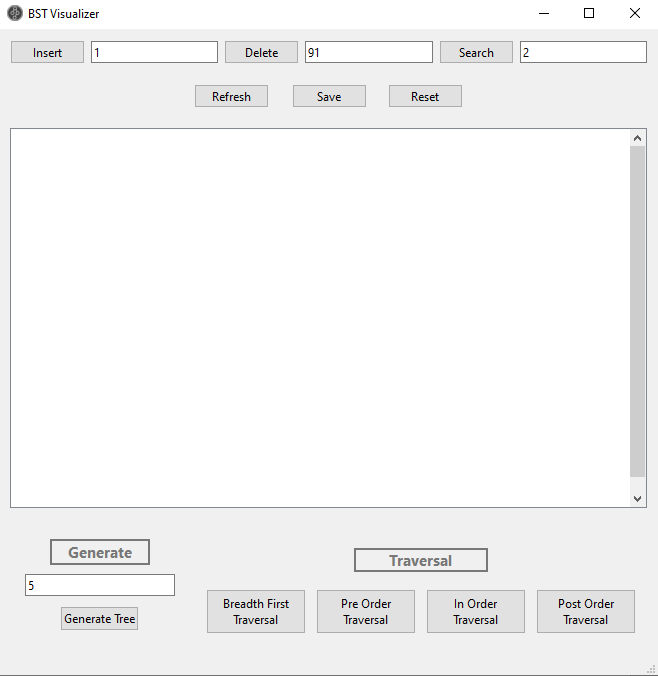
Traversals, insertion, search and delete will usually leave a node with a green color as a leftover from their operations. For your convenience before starting the next operation, it is recommended to refresh the tree first to return it to its initial state (without green-colored nodes) to avoid confusions.



1. Result of execution: Saving the tree



1. Result of execution: Resetting the tree



All executions except refresh, save, and reset will be shown with animation to meet our goals of easing users to learn more about the binary search tree. As of now, the animations goes linearly, meaning that once executed, it cannot be stopped, slowed, fasten, or skipped. The framerate for the animation is 1 frame per second, meaning that it will animate the next node in the tree after 1 second has passed. The demonstration of the animations will be shown in the application demo.

1. **Links**
2. QT installation tutorial and steps to open the project file can be accessed from:

<https://youtu.be/6zgSeZYMjPQ>

1. The application demo can be accessed from:

<https://youtu.be/T4GmJMN8gjQ>

1. The project repository can be accessed from:

<https://github.com/karelbondan/QT_BST_Visualizer>

1. **References**

Feminella, J., fastcodejava, Rode, J. S., & sahil. (2010, February 21). *What does O(log n) Mean Exactly?* Stack Overflow*.* <https://stackoverflow.com/questions/2307283/what-does-olog-n-mean-exactly>

GeeksforGeeks. (2018, November 22). *Complexity of Different Operations in Binary Tree, Binary Search Tree and AVL Tree*. <https://www.geeksforgeeks.org/complexity-different-operations-binary-tree-binary-search-tree-avl-tree/>

Mateusz Kaczanowski. (2014, February 18). *Binary Tree Visualization with Qt4*. YouTube. <https://www.youtube.com/watch?v=uUi4gPsMBtc>

programminghelporg. (2012, June 17). *C++ Tutorial 29 - Reading and Writing to Files – fstream.* YouTube. <https://www.youtube.com/watch?v=Iho2EdJgusQ>

ProgrammingKnowledge. (2013, July 28). *QT C++ GUI Tutorial 5- How to Open a New Window from a Pushbutton in Qt*. YouTube. <https://www.youtube.com/watch?v=tP70B-pdTH0>

ProgrammingKnowledge. (2016, April 14). *Qt Tutorials For Beginners 5 - Qt Signal and slots*. YouTube. <https://www.youtube.com/watch?v=F56fSKoNCtk>

ProgrammingKnowledge. (2020, July 09). *How To Install Qt Creator on Windows 10*. YouTube. <https://www.youtube.com/watch?v=hQSuM1uCdXE>

Russian Engineers. (2015, April 20). *Deployment Qt Apps on Windows.* YouTube. <https://www.youtube.com/watch?v=8qozxqSZQEg>

Wikipedia. (n.d.). *Binary Search Tree.* Retrieved 2021, June 28. <https://en.wikipedia.org/wiki/Binary_search_tree>