Animation Environment for Linked Data Structures – Extended Abstract

Kiu Man Yeung
118100055

Introduction

The basis of this project is to provide data structures visualisation for educational purpose, which students can inspect their code to achieve better learning result. The main goal of the project is to create a software that shows step-by-step execution of computer algorithms on data manipulations in Python. The software provides a graphical interface that allows users to input code implementation and display a stepped animation of the execution beside.

Motivation

Data structures and algorithms is a must-learn topic in a Computer Science or Software Engineering education. Data structures is a particular way of organising and storing data. It describes how data is grouped and linked. It is used in conjunction with corresponding operation algorithms for data management. The main focus of learning data structures and algorithms is correctness and efficiently. Being able to organise data in an appropriate structure enhances the algorithms' efficiency and showcases one's ability in problem solving.

Many students struggle with understanding, choosing, or implementing data structures. Students usually start learning data structures and algorithms after introductory level programming course. Many memory management concepts are very often not yet introduced to students. Students may not have heard of or understand how data linking mechanism, like pointers, works. They may already have experience with canned data structures without know what is going on in the black box.

It requires abstract thinking an imagination to learn data structures. Instead of just reading the code examples and lecture slides, one of the best ways to dive into data structure is to spend time with pen and papers. Drawing and mocking up structures is very helpful for learning the underlying concept. Students will be able see how data structures are constructed and linked. By drawing the structure after every step in the algorithm, it allows students to understand how the operation has modified the data and what it has achieve.

However, the implementation does not behave the same as on paper if there is a mistake. The drawings are not a direct representation of the execution. It is rather an expected result of the execution. The graphs and the implementation are not technically connected and therefore cannot reflect problems of broken data structures and algorithms. This project aims to help students better understand the topic by visualising the structures they have created.

Analysis

Objectives

The goal of the project is to create a software that bridges between data manipulation algorithms and the actual graphical representation of the execution.

The software takes Python code as input from the user and generates graphs of data structures for each step in the execution. The visualisation is displayed next to the source. It is updated as the code is being stepped through. The execution can be paused at any step and travelling back and forth between steps is allowed.

The software is required to handle data structures introduced in CS2515 and CS2516, Algorithms and Data Structures I and II. Including but not limited to:

- Lists
- Stacks
- Queues
- Linked Lists
- Double Linked Lists
- Binary Trees

Feature Requirements

The software must:

1. The software must have a graphical user interface (GUI) for interaction with users.
2. There must be a textbox input for users to enter their implementation.
3. The software must be able to analysis Python code.
4. The software must be able to generate graphs that represents the data structures of the user's implementation.
5. Canvas must update to show the result of current step in the execution.
6. There must be buttons for going back and forth between steps.

The software can:

1. An arrow pointing at the line of code that is being executed.
2. Native canvas for drawing data structures.

The software will not:

1. It will not be taking anything programming language other than Python.

Prerequisites

The software will be developed with Python, using the following libraries and packages: Abstract Syntax Trees (AST), Graphviz, lolviz, tkinter. Python 3, Graphviz and lolviz must be installed prior to running the software.

Design

There are three main components in this software architecture: GUI, parsing, and graph visualisation.

GUI

The GUI toolkit of choice is tkinter. It is an interface to the Tk GUI toolkit that is bundled with standard Python installation on most Unix platforms and Windows ("Python interface to Tcl/Tk", 2022). It is responsible for any communication between the users and the software. It provides an interface where users can enter Python code for analysis and display the results that is generated by the software.

Parsing

AST is a module that helps Python application to process trees of the Python abstract syntax grammar ("ast — Abstract Syntax Trees", 2022). The code from the GUI is passed into this parsing mechanism. It turns the source code into a parse tree. This allows us to insert and trigger graph visualisation after every single statement.

Graph Visualisation

Lolviz is a data structure visualisation tool that can make Graphviz display common Python data structures such as list, dictionaries, linked lists, and binary tress nicely.

Whenever visualisation is triggered, lolviz generates a graph based on the current snapshot of the data structure. The graph is then sent back to tkinter for display.

Main

The main body of the software is the driver that connects all components. It manages setting up GUI, getting input from GUI, pass it to the parser where it triggers visualisation, and feeding the graphs back to the GUI.

Citation

tkinter — Python interface to Tcl/Tk. (n.d.). Python 3.10.2 Documentation. Retrieved March 9, 2022, from https://docs.python.org/3/library/tkinter.html

ast — Abstract Syntax Trees. (n.d.). Python 3.10.2 Documentation. Retrieved March 9, 2022, from https://docs.python.org/3/library/ast.html