Animation Environment for Linked Data Structures – Outline

Kiu Man Yeung
118100055

Linked Data Structures and Algorithms

Data structures are orderly arranged frameworks which are used to maintain data. Linked data structures are a set of abstract data types where they carry reference to another object of the same type. They are high level organisable data structure patterns that allow programmers to manipulate data with algorithms. Correctness and efficiency are two main focuses when learning algorithms around data structures. Being able use data structure correctly and efficiently showcases one's ability in problem solving. In this project, we focus on helping beginners to build algorithms with confidence in correctness.

Obstacle

Learning linked data structures requires imaginations and abstract thinking. Many learners find it helpful to visualise data structures by drawing diagrams to understand how each step in a manipulation works. With a list of notes and links created, drawing an updated graph for each manipulation helps keep track of the structure when modification has been applied.

However, when the algorithm is implemented incorrectly, the program behaves differently than what is expected. Instead of reflecting how the algorithm is actually executed, a hand drawn graph is technically instead a presentation of what the programmer is expecting. There is a disconnection between the algorithm and the graph. It does not have the capability of showing how and where the algorithm may possibly has broken, leaving beginners clueless about the gap between the expectation and the reality. Let alone redrawing the whole structure every step is far from being convenient or efficient.

The Project

This project aims to build a tool that bridges the gap between the algorithm and the graphical visualization. It is specified to analyse Python code. User will be able to see all the objects graphed side by side with the algorithm. The graph updated live with the code executed line by line to demonstrate the effect of the operations. Links are visually shown as arrows between objects. It is expected to show not only the linked objects, but also those that are garbage collected during manipulation to cope with poorly implemented algorithms. Data structures that the tool must be able to address including, but not limited to, Stack, Queue, Linked List, Double Linked List, Binary Tree, etc.

The tool will be implemented in Python. One of the two main component of the tool is parsing. It should allow the tool to give structural representation of the code input. The idea is to have a data structure to keep track of what has been created or modified, even with those that are discarded by the garbage collector due to loss of reference. The other part of the tool is visualisation from the structure that is created by parsing to a graphical presentation. This will be done with the help of external module. It must have the capability to at least show lists and tree structures with pointers.

The next step is to explore some possibly helpful modules for parsing and visualisation. Testing their abilities and limitations to find out what suits the project. It is also important to start planning for the architecture of the program. Further define the requirements and scope of what should, what not and what may be achieved. Create small deadlines to keep the project going and make sure it meets all the deliverables.