Google
Summer of Code

Graph libraries Integration and redesign[350hr]

09.03.2023

—

Mentee: Viren Varma
National Institute of Technology, Karnataka Surathkal

Mentors: Sebastian Jordan, Gordana Rakic
Pharo Consortium

Learning Phase

As part of my learning phase in Google Summer of Code 2023 which was from May 4th to May 27th, I went through all Object Oriented Programming concepts and studied about the pharo IDE. I created 2 packages for practice:

MyCounter Package : Is a simple package with test cases that increments and decrements a class variable `count` using methods increment and decrement.

Binary Heap Package : Is a test package that offers implementation of binary heaps, facilitating efficient priority queues and heap-sort operations within the Pharo programming environment.

AStar Algorithm

Astar Algorithm : I coded out an Astar algorithm with Dijkstra path distance value as a heuristic function. We initially took the number of nodes between 2 nodes as the heuristic value but ended up with going with Dijkstra. I also wrote specific test cases for the algorithm. The algorithm was coded keeping standard API practices in taking references of API design of other similar algorithms.

Longest Path Algorithms for DAG and DCG

Longest Path Algorithm of Directed Acyclic Graph : I used a variation of the Topological Sort algorithm (which is used to find the shortest path in a DAG) to find the longest path. Specific test cases were also written for it.

Longest Path Algorithm of Directed Cyclic Graph : I used a variation of the Bellman Ford algorithm (which is used to find the shortest path in a DCG) to find the longest path. Specific test cases were also written for it.

Redesign Graph Algorithms Test Library

Graph Algorithms Library : Features of Graph visualization of weighted and unweighted graphs have been added to the package of Graph Algorithms testing. The Graph fixtures have also been classified in terms of their presence of weight, cycles or complexity. The Graph fixtures have been classified into the following classes:

- AINonWeightedDAGFixture
- AIWeightedDAGFixture
- AICyclicNonWeightedSimpleGraphFixture
- AICyclicNonWeightedComplexGraphFixture
- AICyclicWeightedSimpleGraphFixture
- AICyclicWeightedComplexGraphFixture

Graph visualization can be viewed upon inspection on Ctrl+I or Ctrl+G. We created a YouTube demo of the visualizations of each type of each graph class.

Link: https://www.youtube.com/watch?v=pocSL1XZy4E

Working on Graph Layouts

The Graph Layouts library (graPharo) had scope for future work before being integrated with the Graph Algorithms library:

Graph Structure and Graph Visualization : Changed Graph Structure to accommodate for accessors and better node traversal. To accommodate and code a graph visualization functionality that can visualize all types of graphs.

Fixing Bugs and adding Test cases : Fixed all bugs in the graph structure and algorithms. And improved current test cases and added new ones to better error catching errors.

Updated Traversing Nodes to O(1)

Node Traversal : Initially nodes were traversed linearly at O(N) time complexity, which seemed inefficient. Several other approaches were tried out related to binary search to optimise the search to O(logN) but they seem to only be effective upto a certain number of nodes. Failed PRs: 1 2

After a few failed PRs, a fresh approach on mapping node labels with nodes using dictionaries was finally implemented.