Shortest Path Algorithms Verification with Idris

Yazhe Feng

February 17, 2019

1 Introduction

Shortest path problems deal with finding the path with minimum distance value between two nodes in a given graph. One variation of shortest path problem is single-source shortest path problem, which focus on finding the path with minimum distance value from one source to all other vertices within the graph. Among all the algorithms that solve single-source shortest path problems, Dijkstra's and Bellman-Ford algorithms are the most renowned, and are implemented by software concerning various fields in real-life applications, such as finding the shortest path in road map, or routing path with minimum cost in networks.

Existing resource on verifying programs that implement Dijkstra's and Bellman-Ford are relatively limited. In most cases the correctness of program relies on the theoretical proof of the underlying algorithm, whereas the verification of program itself remains unattended. Consider the increasing significance of software implementing both algorithms in solving real-world issues, it is important to be able to verify the behaviors and ensure the correctness of these software.

In this work we focuses on verifying simple programs that implement Dijstra's and Bellman-Ford algorithms. Our implementaion starts with defining data structures used in both algorithms(for instance node, edge, and graph), and involves proving properties of data types such as natural numbers and list. We aim to present verification as a programming issue, showing how properties of data types, behaviors of functions, and correctness of programs can be verified not only through theoretical proofs, but also through implementations.

The structure of the paper is as follows. Section 2 describes the significance and value of algorithm verification, and the reason of choosing Idris as the language for verifying programs. Section 3 provides some background on Dijkstra's and Bellman-Ford algorithms, and also briefly introduce the Idris programming language. Section 4 includes an overview of our program, including pseudocode and theoretical proofs of Dijkstra's and Bellman-Ford, which serves as the guildeline for our implementation. In section 5 we will cover more details of our programs, including type signature, function definitions, and key lemmas, presenting a detailed demonstration of our verification program.

2 Motivation

3 Background

3.1 Dijkstra's Algorithm

4 High-level Contribution

5 Literature review

5.1 Some subsection

5.1.1 And a subsubsection

6 Theory

7 Research Design

8 Analysis

9 Conclusion

References

Appendix

Statutory Declaration