Document number: P1449R0
Date: 2019−01−21
Project: ISO JTC1/SC22/WG21: Programming Language C++
Audience: SG19 (Machine Learning Study Group)
Reply to: Vincent Reverdy (vince.rev@gmail.com)

Towards Tree and Graph Data Structures for C++

Vincent Reverdy[1]

[1] *Department of Astronomy, University of Illinois at Urbana-Champaign, 1002 W. Green St., Urbana, IL 61801, USA*
[1] *Laboratoire Univers et Théories, Observatoire de Paris-Meudon, 5 place Jules Janssen, 92190 Meudon, France*

Abstract

In the context of the starting Study Group 19 on Machine Learning, we open the discussion on tree and graph data structures. Trees and graphs have long been needed by C++ applications but given the wide variety of potential customization points, it has often been considered that they would be nearly impossible to standardize. However, as they are essential to many machine learning algorithms, SG19 is bringing the topic back to life. Additionally, ongoing research in high performance computing and generic programming at the University of Illinois at Urbana-Champaign has demonstrated that genericity in trees and graphs is not completely out of reach. Furthermore with upcoming concepts, ranges, and soon reflection, new programming models will be available to achieve at the same time genericity, performance, and ease-of-use. The goal of this paper and its future revisions is to act as a progress report on the topic. This first version just opens the discussion and invites other authors to join the effort.

Contents

1 Introduction

Tree and graph data structures are as fundamental for computing as containers. However, the standard C++ library still do not provide a standard tree and a standard graph. One of the reason is that they come in many forms, with different properties, and constraints. There is also a wide variety of possible underlying memory layouts offering a wide range of performances depending on the platform. And there is also many ways to iterate through them. Without generic programming, there simply is a combinatorial explosions of possibilities. The Boost Graph Library explored some of this complexity. However, concepts, ranges, and reflection will allow new programming models to build on this work and extend it to offer fully customizable standard trees and graphs to the C++ community. In machine learning, one does not need to dig deep to need trees: simple algorithms such as random forests already require them.

2 Tree and graph building blocks

Different algorithms will have different requirements. Also, focusing on one particular type of trees or graphs, such as kD-tree, binary trees, quadtrees, octrees, polytrees, acyclic graphs, hypergraphs would lead to the standardization of hundreds, if not thousands, of different data structures. One way to approach the problem will be to design data structure building blocks instead of the data structures themselves. These building blocks will then be assembled together by application developers to build the perfect data structure for their exact need. Example of building blocks include node type, memory layout, iterator type, traversal strategy, algorithms…The library should allow users to write a simple generic graph implemented as an adjacency matrix, as well as something as exotic as a fixed-maximum-size exponential anti-arborescence implemented as an implicit data structure, with post-order depth first iterators that keep track of node ancestors for an optimized $\mathcal{O}(1)$ access. This effort is envisioned to be a long-term conceptification effort to make tree and graph building blocks fully composable while providing sound customizations points and delivering maximum performance.

3 Collaboration

The author invites all people interested in the topic to participate in SG19. The author plans to update the paper and post revisions from meeting to meeting to keep track of the progress.

4 Acknowledgements

Vincent Reverdy's work has been made possible by the University of Illinois at Urbana-Champaign and by NSF Awards CCF-1647432 and SI2-SSE-1642411.