Software Tool to Calculate the Tree-depth of a Graph

Uddhav Agarwal

Sebastian Ordyniak

COM3610 – Dissertation Project

20/11/2018

The University of Sheffield

This report is submitted in partial fulfilment of the requirement for the degree of Computer Science by Uddhav Agarwal.

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do so amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

-Uddhav Agarwal

Abstract

Graph theory, a relatively newer topic is also one of the most talked about topic in the fields of Discrete Mathematics and Theoretical Computer Science. In addition to the various applications amongst the subjects such as Physics, Chemistry, Biology, Mathematics and Computer Science, graph theory has several other application that span over a range of disciplines such as Linguistics and Social sciences [insert Wikipedia reference]. This preliminary report highlights the problem and significance of calculating the tree-depth for a graph. The final aim of this project towards the end is to develop a software using Python, which implements and computes the tree-depth for a graph. Starting from a basic depth-first search algorithm, an attempt to implement algorithms that are more complex will be made.

Contents

[Introduction 5](#_Toc531185157)

[Definition and Properties 6](#_Toc531185158)

[About the Project 7](#_Toc531185159)

[Literature Survey 8](#_Toc531185160)

# Introduction

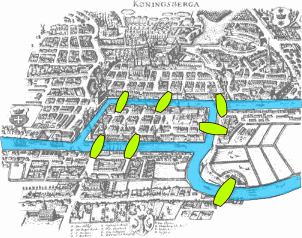
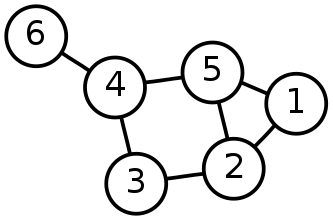
The first paper on graph theory is known to be written by Leonhard Euler in 1736 (Euler, 1736). His aim was to find a solution for the then-famous problem called the Seven Bridges of Königsberg (Figure 1). The problem asked for a path that travels through all four land masses surrounding the river under the condition that the bridges (marked in green in Figure 1) could only be crossed once. Euler suggested that the route on land is irrelevant and only the order in which the bridges are travelled matters. As a result, he considered each land mass as one object or a ‘node’ and the bridges as the paths or ‘edges’ connecting the nodes.

Figure 1. Example of a graph with six vertices and seven edges. Here, V = {1, 2, 3, 4, 5, 6} and E = {{1,2}, {1,5}, {2,5} ,{2,3} ,{5,4}, {3,4}, {4,6}}

Figure 2. An image showing the map of Königsberg, displaying the layout of the Seven Bridges problem

That laid the foundation for the modern day graph theory. In a similar fashion, today we can define a graph G as an ordered pair G = (V, E) where V is a (finite) set of vertices (also known as nodes) and E is a set of 2-subsets (pairs) of V or edges.

Graphs can take many forms based on the structure they have such as a complete graph, bipartite graph, cyclic graph, directed graphs, connected graph, etcetera. These structures can be combined further to form many different combinations of graphs.

Mathematicians and Computer Scientists alike have been interested in graphs and its invariants. Countless problems have been developed to do with graphs such as subgraph isomorphism problem, Hamiltonian path problem, travelling salesman problem, vertex cover problem, etcetera.

This aim of this project is to develop a software tool to solve and calculate one of the parameters of a graph which is a problem on to itself which has seen increasing interest in recent years. That parameter is called the tree-depth.

## Definition and Properties

Before we look into tree-depth we must first define the various components of a graph. A tree is an acyclic-connected graph. A forest can be defined as a collection of trees or a graph where the connected components are trees (Supervisor and Chen, 2015). The height of a vertex x in a rooted forest is defined as number of nodes in the path from the root (of the component x belongs to) to that node x and is denoted by height(x, F). Hence, we can deduce that height of a forest F as a whole is the maximum height of the vertices of F. Next, we look into the ancestors in a graph. Let x, y be two vertices of a forest F. Then x is an ancestor of y if in F, x belongs to the path where y is connected to the root of F. Closure of a rooted forest F or clos(F) is a graph with vertex set V(F) and edges such that {{x, y} : x ≠ y, x is an ancestor of y in F} (Nešetřil and Ossona de Mendez, 2012).

Now that we know some basic building blocks of a graph, we can define tree-depth. Tree-depth is considered an equivalent or similar notion to rank function, the vertex ranking number, the minimum height of an elimination tree, etcetera. (Nešetřil and Ossona de Mendez, 2012) define the tree-depth td(G) of a graph G as the minimum height of a rooted forest F such that G ⊆ clos(F), where clos(F) stands for closure of a rooted forest F (see Figure 3.)

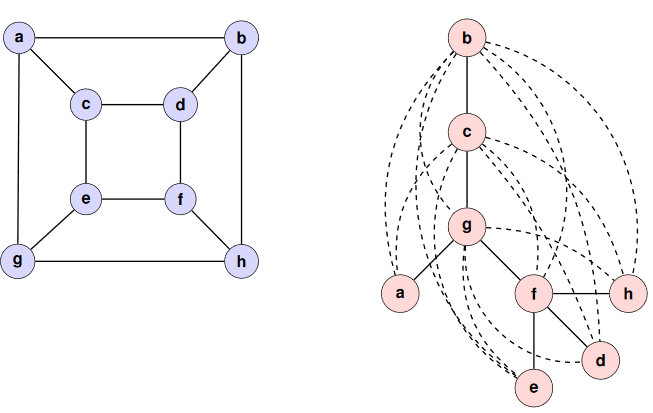


Figure 3. Adopted from (Supervisor and Chen, 2015). The graph G and tree T are on the left and right respectively. The dotted lines in T represent the clos(T). From the definition stated earlier, we know that G ⊆ clos(T) and height(T) = 5, hence td(G) is 5

In addition to the definition above (Nešetřil and Ossona de Mendez, 2012) and (Supervisor and Chen, 2015) provide a similar representation of the tree-depth as a recursive function on the elimination tree of a graph G. The elimination tree F of a graph G with vertex set V(G) defined as follows:

* If V(G) = {v} or |G| = 1, then F = {v}
* If G is connected and |G| > 1, then r ∈ V(G) is chosen as the root of the of F and an elimination forest is created for G – r. The roots of this elimination forest will be the children of r in F.
* Otherwise, if G is not connected, then F is the union of the elimination forest of each component of G

(Nešetřil and Ossona de Mendez, 2012) proves the lemma that tree-depth of a graph G td(G) is the minimum height of an elimination tree for G and provides the following recursive formula (see Figure 4).

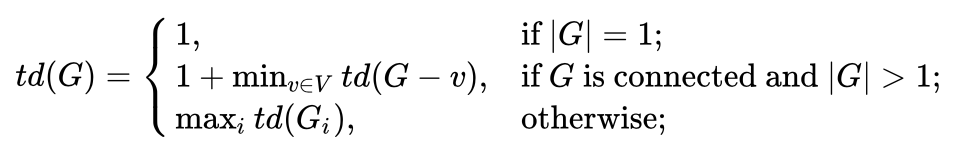


Figure 4. Recursive formula for tree-depth using the elimination forests as devised by (Nešetřil and Ossona de Mendez, 2012). Here, Gi are the connected components of G

## About the Project

This preliminary knowledge about graphs is important to understand the aims and scope of this project. The aim of this project is to develop a software tool using Python and various libraries to obtain a tree-depth decomposition using a simple depth-first search in the graph and then look into the possibility of improving the result by implementing already proposed algorithms.

The language that the software will be written in is Python, as it provides a versatile coding environment, Python is a fast for prototyping, it has a vast array of libraries and also because of my previous experience coding for my other personal and university projects. The libraries and packages used are:

* NetworkX (Hagberg hagberg *et al.*, 2008)
* IPython (Sos, 1989)
* Matplotlib (Hunter, 2007)
* SciPy (Oliphant, 2007)

IPython, Matplotlib and SciPy are the more commonly used libraries and their applications are known and are expected to be known by the reader. NetworkX is a python library used to study graphs and networks, which is the main focus of this report. NetworkX streamlines the process of graph representation and can read/write various graph formats. The nodes in NetworkX can store any Python object and the edges can also store arbitrary data.

For the graph data, the ‘famous’ DIMACS graph format will be used. This format for the undirected graphs was defined by DIMACS (Centre for Discrete Mathematics and Theoretical Computer Science) and has been used as a standard format for undirected graphs. (DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), no date)

When running the software tool, the user will have to specify the file containing the graph data and the algorithm they wish the software tool to implement to find the tree-depth in the Terminal window. The program should return the tree decomposition that was attained with the resulting tree-depth. Just for some added information, the program should also return the ‘retrieval time’ which is the time comparison between the algorithm used and the size of the input graph data.

In the Chapter 2 ahead, I will discuss the review of various journals and papers that talk about possible heuristics to improve the time complexity of obtaining the tree-depth decomposition and hence finding the tree-depth of the graph. In chapter 3, I will provide the details of my aims and objectives with the project, which includes the parts covered by the project and the scientific evaluation. In chapter 4, I will discuss my progress so far and briefly describe my roadmap. Chapter 5 will be a conclusion of this preliminary report followed by the bibliography.

# Literature Survey