**SD LAB 6: Graphs, trees & Containers.**



Student: Johnson Domacasse

Date: 27 Nov 2023

Student#: 4471709

Teacher: Lake Lakeman

Contents

[Introduction: 3](#_Toc153545330)

[Binary trees: 3](#_Toc153545331)

[Graphs and Containers: 3](#_Toc153545332)

[References: 3](#_Toc153545333)

# Introduction:

*The purpose of this document is to show or justify the decisions that were made during this assignment. Below you will find the sections that talk about trees and graphs. Inside these sections there will be the design choices I made, along with a reflection on these design choices.*

# Binary trees:

For this implementation, We focus on two things. The algorithm to find the depth of the binary tree**[1]**, and the way we create the tree, based on the input files. For this implementation.

For the algorithm**,** I have decided to use the Depth-First search algorithm (DFS) over the breadth-first search algorithm (BFS) for the following reasons**[3]**:

* **“DFS is more suitable when there are solutions away from the source”:** In this case we are trying to find the depth of a binary tree, we don’t know whether the depth is one level lower or hundreds.
* **“DFS requires less memory than BFS”:** Memory isn’t a big problem in this case but the less we use the better I would say
* **“when the target is far from the source, DFS is preferred”:** Again, BFS could’ve been a better option in this case if the depth level was fairly low. However since we do not know if the depth is a level lower, or hundreds, we use DFS.

With that being said a DFS algorithm function was implemented**[4].**

For reading the input file, a different approach was taken. Reading the user input was easy, we would also need to store this information somehow. The following drawing shows the thought process:

A piece of paper with writing on it

Description automatically generated

From here, the tree was drawn, where it was realized that the nodes needed to be saved in some type of container. From there the options were weighed. This was implemented using vectors because of their ability to dynamically resize themselves during run time. Initially the approach was using an array because the size of the tree is already given. This was to get some practice in with vectors. This implementation gives us the option to have our storage grow dynamically in an efficient way at the cost of memory**[5].**

# Graphs and Containers:

# References:

*This section is to give all of the references that I have used in the making of this assignment.*

**[1]** – GeeksforGeeks. (2023e, September 27). *Binary tree data structure*. GeeksforGeeks. <https://www.geeksforgeeks.org/binary-tree-data-structure/>

**[2]** – MyCodeSchool. (2014, January 18). *Data Structures: Binary Tree*. MyCodeSchool. <https://www.youtube.com/watch?v=H5JubkIy_p8&t=838s&pp=ygULYmluYXJ5IHRyZWU%3D>

**[3]** – GeeksforGeeks. (2023, October 3). *DFS & BFS Algorithms*. GeeksforGeeks. <https://www.geeksforgeeks.org/find-the-maximum-depth-or-height-of-a-tree/>

**[4]** - GeeksforGeeks. (2023b, October 3). *DFS in binary tree*. GeeksforGeeks. <https://www.geeksforgeeks.org/find-the-maximum-depth-or-height-of-a-tree/?ref=lbp>

**[5]** - *Vectors*. cplusplus.com. (n.d.). <https://cplusplus.com/reference/vector/vector/>

**[6] -** GeeksforGeeks. (2023a, June 9). *Breadth first search or BFS for a graph*. GeeksforGeeks. <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>