Graphs, trees and recursion

Assignment documentation by

Veselin Atanasov

Teacher: Lake Lakeman

Date:15-01-2025

Assignment 1 (recursion)

The goal of this part of this assignment is to create an algorithm that finds the depth of a binary tree. The two types of algorithms to search through a binary tree: BFS(breadth-first search) and DFS(depth-first search). BFS is a graph traversal algorithm that explores nodes level by level, starting from a given source node. It uses a queue to process nodes, beginning by marking the source node as visited and enqueuing it. At each step, the algorithm dequeues the current node, visits all its unvisited neighbors, marks them as visited, and enqueues them. This process continues until the queue is empty, ensuring that nodes are explored in increasing order of distance from the source. DFS is a graph traversal algorithm that explores as far as possible along each branch before backtracking. Starting from a given source node, DFS marks it as visited and recursively visits its unvisited neighbors one by one, diving deeper into the graph. This process continues until all paths from the current node are fully explored, at which point it backtracks to the previous node to explore other branches. Unlike BFS, which uses a queue, DFS typically uses a stack (either explicitly or via recursion) to manage the traversal. DFS does not guarantee the shortest path in a graph but is effective for tasks like detecting cycles, finding connected components, or solving maze problems.

For both of the assignments I decided to use algorithms based on BFS as it is easier to understand and also fits the criteria of assignment 2 for finding the shortest route to a node in a graph.

Since I am using algorithms based on BFS and BFS utilizes vectors and queues, I also decided to use the same data structures. An addition I made is a struct called Node which has members left and right to store the neighbours. This struct is stored in a vector. The index of the vector works as a root of the node and the members of the node struct point to the neighbours of this node. Then a queue of integers is created to store the unchecked nodes. The integer 1 is pushed into the queue to start the search from the first node. After this the integer is popped from the queue and a check is done to determine whether it has neighbours. If it does, they are added to the queue, if there aren't, nothing will happen. After this process, the depth variable is updated to show that a new level of depth has been reached. This is done over and over again until the queue is empty. If the queue is empty, that means that the end of the binary tree has been reached and the total depth is reached.

Assignment 2 (graphs, shortest-path)

For this task I reused a BFS algorithm from the internet. This algorithm pretty much works in the same manner. The key difference is that in graphs there can be a bidirectional connection which may lead to visiting a node which has been visited already. To avoid this, a vector of Booleans is created to keep track if a node has been visited or not.

Sources

https://medium.com/basecs/going-broad-in-a-graph-bfs-traversal-959bd1a09255

https://wikipedia.org

https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/