CMPT 435 - Spring 2019 - Dr. Labouseur

Assignment Three – Graphs & Trees

Christopher Ravosa
christopher.ravosa1@marist.edu

April 17, 2019

1 Running Time of Breadth-First Traversal

In a breadth-first traversal, the exploration status of each vertex is maintained in a queue. The traversal begins with the starting vertex being added to the queue. The traversal then takes note of each of the vertices connected by an edge to the starting vertex, adding each adjacent vertex to the queue. The starting vertex is then noted as having been analyzed and is removed from the queue.

The traversal then takes note of each adjacent edge of each of the remaining vertices in the order they are enqueued, dequeuing each when it has finished. This is where the name "breadth-first" comes from, because the function analyzes the breadth of each vertex's adjacency list before going to the next level of vertices represented in the queue.

Ignoring vertices it has already visited, a breadth-first traversal continues recording each vertex in the adjacency list of each vertex it dequeues. This occurs until the queue is empty and no vertices have been left unmarked.

To clarify, a breadth-first traversal interacts with every vertex transitively connected to the starting vertex, and every edge thereof, exactly once. Therefore, the asymptotic running time of a breadth-first traversal is O(|V| + |E|); where 'V' equals the number of vertices in the graph, and 'E' equals the number of edges.

2 Running Time of Depth-First Traversal

Though the task of a depth-first traversal is similar to that of a breadth-first traversal, it is a different function entirely. In contrast to a breadth-first traver-

sal, a depth-first traversal is typically a recursive function with two necessary methods.

The first method in a DFT is very straightforward. It simply iterates through the graph, calling its recursive helper method on every vertex that has not yet been interacted with. The recursive method, referred to in the textbook as "DFS-Visit", is where most of the lifting occurs.

DFS-Visit takes both the graph and the untouched vertex as parameters. It begins by setting the vertex as marked so that it doesn't get touched again. Next, the helper method iterates through the list of vertices that are adjacent to the vertex being analyzed. If an adjacent vertex has not been interacted with, DFS-Visit is recursively called on that vertex. After inspecting the current vertex's edges, and the vertices on the other ends, the starting vertex is marked as completed. Like in a breadth-first traversal, the marking process prevents vertices from being manipulated more than once.

While a breadth-first traversal must maintain a queue in order to function properly, a depth-first traversal requires only pointers to perform. Therefore, a depth-first traversal is more space-efficient than a breadth-first traversal; however, its running time remains the same. Like a breadth-first traversal, a depth-first traversal also interacts with every connected vertex and edge exactly once. Therefore, a depth-first traversal shares with breadth-first traversals, a running time of O(|V| + |E|).

3 A Note on Marking Vertices

It is important to mention briefly that the "marking" action that is performed on vertices in both the breadth-first and depth-first traversals is typically done using color-coding. Each vertex is given a variable to store its hypothetical color, which frequently begins with "white". The first time the vertex is acknowledged by either function, it is changed to another color like "gray". The final time the functions interact with a vertex, the vertex is changed to a final color, like "black", to show that the vertex requires no further action.

4 Running Time of Look-Ups in a Binary Search Tree

A binary search tree is a simple structure and thus, requires simple algorithms to be traversed. The treeSearch function accepts two parameters, being a starting node and a target value. Since each node in a binary tree has at most two children, one less than and one greater than the parent, the tree can be traversed using basic if-else statements. If the value at the starting point is not equal to the target value, then the traversal can either climb down the tree on the left or right path at any given node. Due to this, the traversal traces a simple path from the starting node down until it finds the desired value, encountering only basic comparison functions along the way.

The traversal only encounters one comparison at any given level of the tree, and the lowest the traversal can go is to the bottom of said tree. As a result, in the worst case scenario, a search through a binary tree can only be as long as the tree is tall; therefore, the asyptotic running time of a look-up in a binary search tree is O(h), where 'h' is equal to the height of the tree.