William Covington

Caesar Mier

CPSC 335-05

Project 2: Asgwilanga Caverns

Team: WOC

Project #2 — Searching the Great Asgwilanga Caverns

This document analyzes the Big O running time for Searching the Great Asgwilanga Caverns Algorithm. This program is an algorithm visualization task where we display the steps and progress from the outermost cave to its "most-central" cave room. These caverns have several cave room subsystems that make it longer to reach our desired destination which may affect our Big O running time. The cave rooms each direct you in only one direction. Using a depth-first search we explore the cave using a specific starting color. We explore each cave and only learn about that cave once we explore it and display its data as we go.

Searching the Great Asgwilanga Caverns has a Big O runtime of O(V+E), V for the vertices, and E for edges explored. This runtime is because the number of iterations is dependent on the number of each vertex and edge in the search. The search starts at the root and works its way down exploring the next node on its path. If the search comes to a dead end, it backtracks to the previous node and explores that nodes' unvisited nodes. The search continues until the whole graph's nodes are visited. At most all the vertices and edges get visited only once. It's easy to see why this would be the Big O running time if we consider we are counting the number of vertices and edges in a graph that must be visited.

The space complexity for Searching the Great Asgwilanga Caverns algorithm is O(V) because we need to store the vertices we have explored. We must store each node we explore because we must remember where to return in case we meet a dead end. By storing the previous node, we save our place where we can explore from next. So, we store the nodes in a stack to help with our depth-first search. We do not store any of the edges, so we do not account for them in the space complexity of our algorithm.

In conclusion, Searching the Great Asgwilanga Caverns has a Big O runtime of O(V+E) because the number iterations that get performed is equal to the size of vertices and edges in our graph. The search starts at the root node and proceeds to visit its neighboring node. It continues doing this until we reach a dead end. Then the algorithm backtracks to its previous node and continues to search for unexplored nodes. It does this until the whole graph is visited. The space complexity of the algorithm is O(V) because no additional memory is needed other than storing the nodes in a stack to keep track of what we have explored. This data structure is essential to our algorithm working as we need to remember what nodes we have visited so we can return our search to previous nodes if we reach a dead end.