The shortest path is the least number of hops from one node to another in the graph, if a path exists. As it turns out, the breadth-first search algorithm can be used to determine the shortest path. After each list of adjacent nodes to the current node is added to the adjacency list, we insert a marker. That marker indicates the end of a horizontal row in the graph (note: horizontal is a descriptor for all nodes equidistant from the current node. As we encounter and count markers removing the next node to visit, we have the path length to that target node. When we find the node we are searching for, the path length is known and that is the shortest path.

For example, consider the shortest path from the BFS previously:

Table

Description automatically generated with medium confidence

If we are searching for 11, our path length at the adjacent nodes of 1 would be 1. When we encounter the ~ after 4, we increase the path length to 2.The ~ after eight would cause our path length to increase to 3. At this point we will encounter 11 before another ~, therefor the path length from node 1 to node 11 is three. If we use the DFS algorithm, we would not be able to stop when we encounter 11 because there may be other paths to 11 that are shorter than the one we are currently investigating. Consider this slightly modified example:

Diagram

Description automatically generated

Since the length of the shortest path from 1 to 11 is 3 (through node 8) BFS would return that depth as soon as 11 was encountered (as an adjacent to 8). A DFS algorithm would find 11 from 12 before finding 11 from 8, thus if we don't find all paths, DFS might report 1 to 11 shortest path as a length of 4 when in fact it should be 3.