addNode()

This function first checks if the node is in the graph already. Therefore, addNode() has a complexity of O(n).

addEdge()

This function checks if both nodes are in the graph and their locations in the same loop. Therefore, the complexity of addEdge() is O(n).

dijikstra()

dijikstra() performs the following actions:

- Checks if both nodes exist. O(n)
- Obtains the source node. O(n)
- Loops until queue is empty. O(n)
    - Determines which index has the lowest cost. O(n)
    - Add or update children. O(n)
        - Check if a child is visited. O(n)
        - Check if a child is in the queue. O(n)
- Construct the output path. O(n)

One list stands out as being deep.

- Loops until queue is empty. O(n)
    - Determines which index has the lowest cost. O(n)
    - Add or update children. O(n)
        - Check if a child is visited. O(n)
        - Check if a child is in the queue. O(n)

As the loop is 3 levels deep, this means dijkstra() has a complexity of O(n^3).

prim()

prim() is also 3 loops deep as it is directly based off of dijikstra(). Therefore, it also has a complexity of O(n^3).