COMP3121 21T2 Assignment 3 Q5

Written by Zheng Luo (z5206267)

In order to find a path in graph G of length exactly K that has maximum total weight, both starting vertex and current vertex need to be considered as variables, since graph can be started at any vertex, hence a while loop for all starting vertex needs to be created, as well as a while loop inside of previous loop needs to be implemented for checking all possible paths at specified starting vertex. A dp table needs to be created in order to record the maximum total weight at current position $dp[starting][currentPosition]$. The next position of vertex can be all possible directed neighbours including visited vertex and current vertex itself. In the end, the function will return the largest number in dp table, which will be the maximum total weight in graph G. The actual path can be traced back from maximum value to its previous vertex until it reaches starting vertex. The time complexity is $O(n^2)$ overall, since two while loops have been used to record dp value. The pseudo code below can explain the implementation in more detailed manner.

```
int maxTotalWeight(Graph[V][E] G) {
    // Creating a dp to store max weight at current position so far.
    dp[startingV][currentV];
    // Considering all starting vertex.
    for (Vertex startingV : V) {
        int length = 0;
        List[Vertex] possibleNeighbourList = getAllPossibleNeibours(startingV);
        // Considering all possible neighbours as next vertex,
        // including visited neighbour and current vertex itself.
        for (Vertex currentVertex : possibleNeighbourList) {
            // Ending condition:
            // end the current path if reaches max length K.
            if (length == K) {
                break;
            }
            // filling up the dp table.
            dp[startingV][currentVertex] = Math.max(previous dp + weightOnCurrentVertex);
        }
    }
    // The largest number in dp table is the result.
    return Math.max(dp[][])
}
```