Roll: 32 (Jahir Sadik Monon)

Fundamentals of Genomics and Proteomics Lab (Task 1)

Department of Computer Science and Engineering, University of Dhaka

Problem statement: Implement an algorithm to find a de Bruijn graph for the sequence GACTTACGTACT with k= 3 and generate the corresponding Eulerian walk.

```
import copy

# To make the de bruijn graph from input sequence and k-mer length
def de_bruijn_graph_maker(sequence, k):
    # Create the de Bruijn graph
    graph = {}
    for i in range(len(sequence) - k + 1):
        kmer = sequence[i:i+k]
        # find left and right child
        prefix = kmer[:-1]
        suffix = kmer[1:]
        if prefix not in graph:
            graph[prefix] = []
        graph[prefix].append(suffix)
    # print graph in terminal
    print("Graph adjacency list: ")
    for node in graph.keys():
        print(f"{node}: {graph[node]}")
    return graph

# return the eulerian walk as an edge list by taking de bruijn graph as input
def give_eulerian_path(graph):
    # Find the Eulerian walk
    stack = []
    circuit = []
    current_vertex = list(graph.keys())[0]
    stack.append(current_vertex)
    while stack:
        v = stack[-1]
        if len(graph[v]) == 0:
            circuit.append(v)
            stack.pop()
        else:
            ov = graph[v][-1]
            graph[v].pop()
            stack.append(ov)

    circuit.reverse()
    eulerian_walk = circuit

    return eulerian_walk

# visualize the de bruijn graph and eulerian walk
def visualize_de_bruijn_graph(graph, eulerian_walk_edgelist):
    try:
        import matplotlib.pyplot as plt
        import networkx as nx
        G = nx.MultiDiGraph()
        edgeno = 0
        for i in range(0, len(eulerian_walk_edgelist)):
            edge = eulerian_walk_edgelist[i]
            G.add_edge(edge[0], edge[1], rad = eulerian_walk_edgelist[0:i].count(edge)/10,
                       color = "gray")
```

```
        pos = nx.planar_layout(G)
        nx.draw_networkx_nodes(G, pos)

        for edge in G.edges(data=True):
            nx.draw_networkx_edges(G, pos, edgelist=[(edge[0],edge[1])],
                connectionstyle= f'arc3, rad = {edge[2]["rad"]}',
                 edge_color = edge[2]["color"])

        edge_labels = {}
        for i in range(0, len(eulerian_walk_edgelist)):
            if eulerian_walk_edgelist[i] not in edge_labels.keys():
                edge_labels[eulerian_walk_edgelist[i]] = f"{i+1}"
            else:
                edge_labels[eulerian_walk_edgelist[i]] += f", {i+1}"

        nx.draw_networkx_labels(G, pos)
        nx.draw_networkx_edge_labels(G, pos, edge_labels)
        plt.show()
    except ModuleNotFoundError:
        print("Install both matplotlib and networkx libraries to visualize!")


# input sequence
sequence = "GACTTACGTACT"
# k-mer length
k = 3
graph = de_bruijn_graph_maker(sequence, k)
visualization_graph = copy.deepcopy(graph)
eulerian_walk = give_eulerian_path(graph)
print("Eulerian walk:", " -> ".join(eulerian_walk))


eulerian_walk_edgelist = [(eulerian_walk[i], eulerian_walk[i+1])
                          for i in range(0, len(eulerian_walk)-1)]
print(eulerian_walk_edgelist)
visualize_de_bruijn_graph(visualization_graph, eulerian_walk_edgelist)
```