The University of New South Wales - COMP9312 - 22T2 - **Data Analytics for Graphs**

Project Homepage / Q1

Q1 (15 marks)

Design a solution to answer shortest distance queries in large graphs.

Problem Statement

In this question, you need to design a solution to compute shortest distance between two query vertices in a **large directed weighted graph**.

Background & Marking Criterias

Given an input graph, you are expected to preprocess the graph and build some index structure to speed up shortest distance query processing for an arbitrary pair of vertices. You can do anything you like to preprocess the graph data.

The marking tutors will evaluate the goodness of your soluton. A good solution should consider query processing time, index space, and index construction time. For example, you can just implement the dijkstra algorithm. There is no index space and indexing time cost, but the query efficiency is low. In the other extreme example, you can just precompute the shortest distance values for all possible pairs of query vertices. The query time is optimal, but the index space is not acceptable since we are dealing with big graphs. By big graphs, we mean graphs with at least millions of vertices. You can think about how much memory (RAM) we need to store shortest distance for all pair of vertices if there are around 10 million vertices. Therefore, the goal is to **improve the query efficiency** while **the additional space usage is acceptable** for million-scale graphs in a commercial machine. The index construction efficiency depends on the structure of your index. Try to design an efficient algorithm to compute your index.

> ### Online VS Offline
>
> We focus on the static graphs without any updates. Normally, the offline (index-based) method is preferable in this setting. Even though preprocessing the graph takes much more time compared with a single shortest distance query processing, there may exist a considerable amount of queries, which can benefit from the precomputed index. If you have no idea about how to design an index-based

> solution, you should at least try to implement a good online (do not preprocess the graph) algorithm .

In addition to writing out the code, it is also important to understand what you have achieved. You need to provide a **document** to describe your solution. The document helps us understand your code, and let us know you have already got an idea to solve the problem at least, which may provide you some marks even if your code cannot run correctly. The document can be in any format. The content should include but not limited to your **designing ideas**, **theoretical analysis for index space complexity, query time complexity, and indexing time complexity**. Example figures will be helpful if you design some complex solutions. You are also welcome to discuss multiple potential solutions/baselines and demonstrate why your solution is the best.

Doing this Project

Open the code template file Q1.ipynb and make a copy of the file in your own google drive. You need to implement a class named `ShortestDistance` with at least two functions, `preprocess()` and `query()`. Below is the code template for `ShortestDistance` shown in Q1.ipynb. You can also find the input graph data structure defined in the `DirectedWeightedGraph` class and an example about how we test your code. The file is running on the Google Colab plarform, which has already integrated the Python environment. As shown in our tutorials, you can directly write and run your code without any configuration. You can directly add any description for your solution and theoretical analysis in your your Q1.ipynb instead of creating a seperated PDF report. Ask your tutor if you have any difficult to run the code.


```
##########################################################################
# Add any modules you want to use here~
##########################################################################

class ShortestDistance(object):
    def __init__(self, G):
        self.G = G
        self.preprocess(G)
        ###################################################################################
        # TODO: You may add some index data structure here~
        # analyze the space usage of the index (all additional data structure)~
        ###################################################################################

    def preprocess(self, G):
        ###################################################################################
        # TODO: Your code here~
        # precompute any data structure for G and use that to speed up your query pr
        # analyze the time complexity of preprocess()
        ###################################################################################
```

```
        return

    def query(self, source_vertex, target_vertex):
        shortest_distance = 0
        ###########################################################################
        # TODO: Your code here~
        # Input: vertex1, vertex2
        # Output: the shortest distance between source_vertex and target_vertex
        # analyze the time complexity of query()
        ###########################################################################

        return shortest_distance

###############################################################################
# You can define any auxiliary functions~
###############################################################################
```

Required Files

Compress all Q1 related files (`Q1.ipynb` and an optional `Q1.pdf` if you want create a documet in a seperated file) into `Q1.zip`, and submit it on Moodle.

Notes

a. A detailed report is required to describe your algorithm and analysis the time complexity of query and index construction and space complexity of the index. You can directly write them in `Q1.ipynb` or in a separated `Q1.pdf`.
b. Any reasonable algorithm is acceptable as long as you have a detailed description and analysis in the report.
c. You can add additional variables and functions in the `DirectedWeightedGraph`. Do not change the input/output format in the code template.
d. We will use a different and much larger dataset to test your algorithm.

END OF QUESTION