The University of New South Wales – COMP9312 – 24T2 – **Data Analytics for Graphs**

Assignment 1

Graph Storage and Graph Traversal

Summary

Submission	Submit an electronic copy of all answers on Moodle (only the last submission will be used).
Required Files	A `.pdf` file is required. The file name should be `ass1_Zid.pdf`
Deadline	**9pm Friday 21 June** (Sydney Time)
Marks	**30** marks (**15%** toward your total mark for this course)

Late penalty. 5% of max mark will be deducted for each additional day (24hr) after the specified submission time and date. No submission is accepted 5 days (120hr) after the deadline.

START OF QUESTIONS

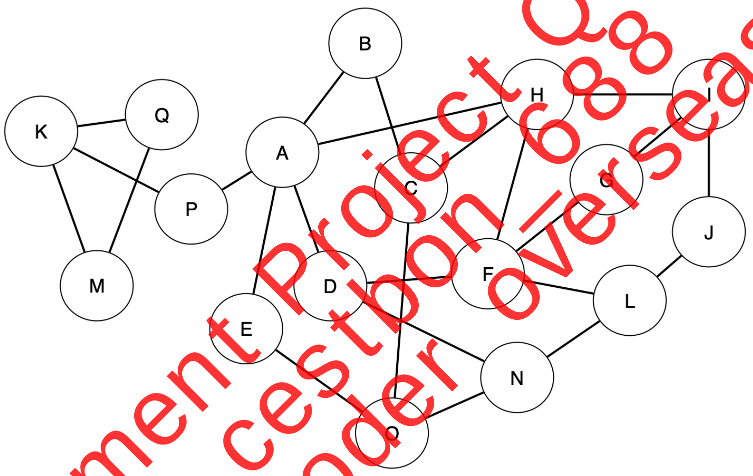

Figure 1

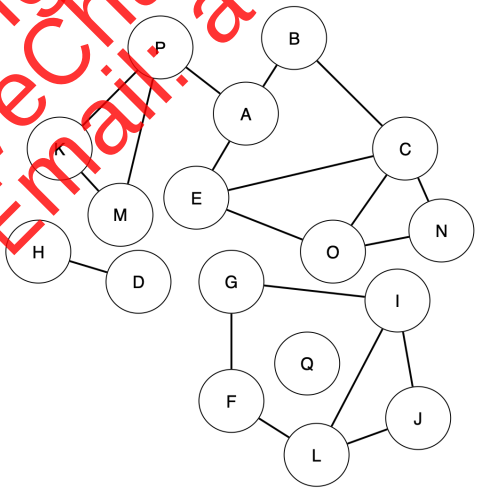

Figure 2

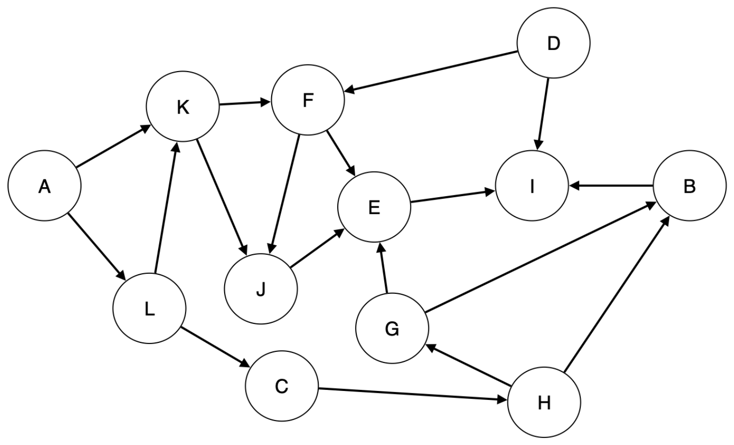

Figure 3

Q1. Required knowledge covered by Topic 1.1 (4 marks)

Please determine whether the following statements for the graph in Figure 1 are TRUE or FALSE.

a. In some correct BFS traversal starting from H, M can be traversed before N.
b. In some correct DFS traversal starting from I, E can be traversed before A.
c. In any correct DFS traversal starting from E, G must be traversed after F.
d. In any correct BFS traversal starting from K, L must be traversed after H.
e. In any correct BFS traversal starting from A, N must be traversed before G.
f. In any correct DFS traversal starting from P, A must be traversed before Q.
g. In some correct DFS traversal starting from M, Q can be traversed after K.
h. In some correct BFS traversal starting from J, A can be traversed after E.

Marking for Q1: 0.5 mark is given for each correct TRUE/FALSE answer.

Q2. Required knowledge covered by Topic 1.1 (5 marks)

Consider the undirected graph in Figure 2 stored by the adjacency list. For each vertex, the neighbors are arranged alphabetically (e.g., the neighbor list of A is [B,E,P]). Describe an algorithm to compute all connected components using the disjoint-set data structure. Show the tree structure after each `union` operation.

Marking for Q2: Full marks are given if each intermediate disjoint-set tree structure is correct.

Q3. Required knowledge covered by Topic 1.1 (5 marks)

Consider the directed graph in Figure 3 stored by the adjacency list. The neighbors of each vertex are arranged alphabetically. Compute the topological order of vertices in the graph. Show intermediate steps.

Marking for Q3: Full marks are given if the described process of each vertex is correct and the order of vertices are correct.

Q4. Required knowledge covered by Topic 0 (6 marks)

We consider an undirected, unweighted graph with n vertices and m edges. Design a data structure to store the graph that can efficiently support the following three operations:
1) Scanning all neighbours of a given vertex,
2) Inserting a new edge that does not exist in the original graph
3) Deleting a vertex from the graph, including all edges related to it.
Justify the time complexity of each operation and the space complexity of the data structure.

Marking for Q4: Two factors are evaluated in marking: (1) How good is your time complexity and space complexity; (2) Does your algorithm match your time complexity. (3) Does your data structure match your space complexity. Full marks are given if your time complexity is not larger than our expected one and your algorithm corresponds with your time complexity.

Q5. Required knowledge covered by Topic 1.1 (5 marks)

We consider an undirected, unweighted graph with n vertices and m edges organized using an adjacency list. Design an algorithm to determine whether there exists a cycle that contains the given query vertex (i.e., the input is a vertex ID, and the result should be TRUE or FALSE). Please write your code in pseudocode and justify the time complexity of each subpart, as well as the total time complexity of your algorithm.

Marking for Q5: Two factors are evaluated in marking: (1) How good is your time complexity; (2) Does your algorithm match your time complexity. Full marks are given if your time complexity is not larger than our expected one and your algorithm corresponds with your time complexity.

Q6. Required knowledge covered by Topic 1.1 (5 marks)

We consider a directed, unweighted graph stored by the adjacency list (an array of out-neighbors is stored for each vertex). Design an algorithm to compute the shortest distance between two query vertices (i.e., the input is two vertex IDs, and the output should be the shortest distance). The queue data structure is not allowed in your solution (e.g., the dequeue object in Python). Please write your pseudocode and justify the time complexity of each subpart, as well as the total time complexity of your algorithm.

Marking for Q6: Two factors are evaluated in marking: (1) How good is your time complexity; (2) Does your algorithm match your time complexity. Full marks are given if your time complexity is not larger than our expected one and your algorithm corresponds with your time complexity.
END OF QUESTIONS