CS2040: Data Structures and Algorithms

Tutorial Problems for Week 10: Graphs and Traversal 1

For: 24 October 2024, Tutorial 8

Problem 1. Bipartite Graph Detection

A *bipartite graph* is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V; but there is no edge between vertices in U and also no edge between vertices in V.

Given an undirected graph with n vertices and m edges, we wish to check if it is bipartite.

Describe the most efficient algorithm you can think of to check whether a graph is bipartite. What is the running time of your algorithm?

Problem 2. Cycle Detection

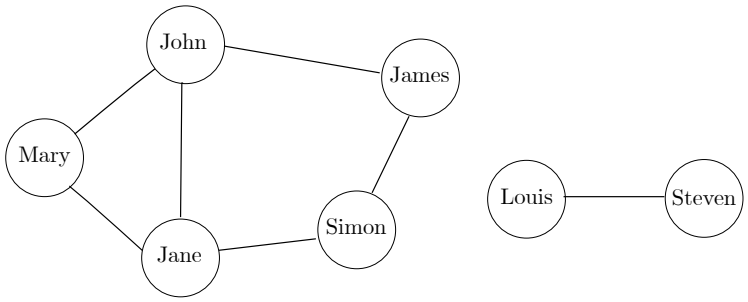
Given a graph with n vertices and m edges, we wish to check if the graph contains a cycle.

Problem 2.a. First, consider the case of an undirected graph. Describe an algorithm to check if the graph contains a cycle.

Problem 2.b. Next, consider the case of a directed graph (with no bi-directed edges). Describe an algorithm to check if the graph contains a cycle.

Problem 3. Friends Network (CS2010 AY17/18 Sem 1 Final Exam)

Peter is a very friendly person who has a lot of friends. In fact, he can construct a graph with n vertices and m edges, where the vertices represent his friends and the edges represent friends who know each other directly. An example is given below.



First, Peter wants to find out if a given pair of friends X and Y know each other directly (e.g John and Jane in the example).

Problem 3.a. What is the most appropriate graph data structure to store his friends graph in this scenario?

Problem 3.b. How would he answer his query using the graph data structure you have proposed in Problem 3a?

Next, Peter wants to know if a given pair of friends X and Y are related to each other indirectly, that is, there is no edge from X to Y but there is at least one path from X to Y with more than 1 edge (e.g. Mary is related indirectly to Simon through Jane among other possibilities in the example).

Problem 3.c. What is the most appropriate graph data structure to store his friends graph in this scenario?

Problem 3.d. Describe an algorithm that answers his query using the graph data structure you have proposed in Problem 3c. What is the running time of your algorithm?

Finally, Peter wants to answer k queries of whether two given friends X and Y are related to each other *indirectly*.

Problem 3.e. Describe the most efficient algorithm you can think of to answer the k queries. What is the running time of *each* query?