Week 13: Part I - Graph Search Algorithms

PROGRAMMING PROJECT:

Finding Connected Components of a Graph

This project is concerned with finding the connected components of an undirected graph. A graph is said to be *connected* if there exists a path in the graph between every pair of nodes. If a graph is not connected, then each connected piece of the graph is called a connected component.

Consider the graph of Figure 1. The subgraphs (a) and (b) shown in the Figure constitute the connected components of the graph.

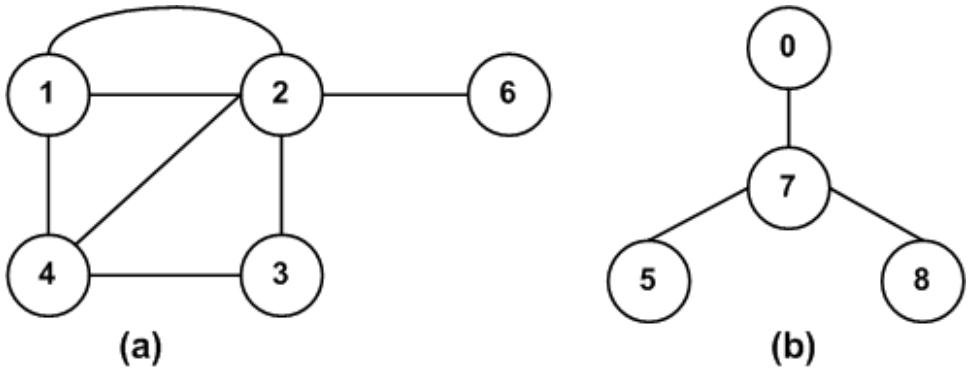

Figure 1: An example disconnected graph

Input Specification: For each graph, the input will be a list of edges of the graph as node-pairs. For example, the graph shown in Figure 1 would appear as a set of the following unordered node pairs (except the first line in which the first integer represents the number of nodes and while the second integer value represents the number of edges respectively):

9 10
1 2
1 2
1 4
2 4
2 3
4 3
6 2
5 7
8 7
0 7

Task: Write a **C++** program to take any input as described above from the file "infile.dat". and perform the following:
(a) Produce the adjacency list of the input graph.

Week 13: Part I - Graph Search Algorithms

(b) Use the algorithm "Explorefrom(s)" (discussed in this week's reading material) to find the connected components of the input graph. (Example: component 1 = (1, 2, 3, 4, 6) and component 2 = (0, 5, 7, 8)).

(c) Write two versions of code and obtain the output:

(i) Version I uses a stack to implement S'.

(ii) Version II uses a queue to implement S'.

Use outfile.dat file to save the output of your program.