Practical Exercise 8 – Graphs

Overall Objective

To design and implement applications using graphs.

Background

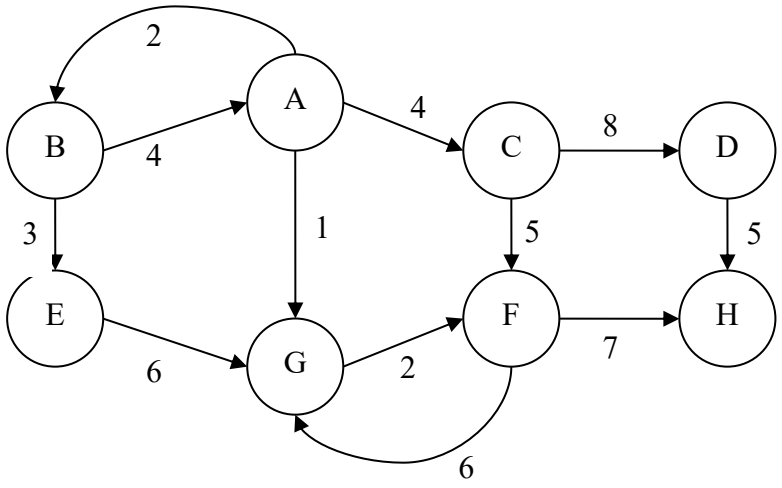
You will need to know:

1. basic Java programming knowledge
2. classes and interfaces
3. generics
4. graph concept

Description

Part 1: Discussion

1. The following graph are given:



1. Construct an ***adjacency matrix*** representation for the graph given above.

2. Construct an ***adjacency list*** representation for the graph given above.

3. Determine whether there is any ***cycle*** or ***loop*** in the graph given above.

4. Start at A, trace a ***depth-first traversal*** through the above graph. You are required to show the ***stack*** contents as you work your way down the graph and then as you back out.

5. Start at A, trace a ***breadth-first traversal*** through the above graph. You are required to show the ***queue*** contents as you work your way down the graph.

6. Start at A, develop a ***minimum spanning tree*** using Prim's algorithm for the graph.

7. Suppose the source vertex is A, develop a ***shortest path*** for the graph using Dijkstra's algorithm.

Part 2: Programming Exercise

Graphs
Refer to lecture slide, define the following interface and classes for the graph:

1. Define `Graph<V>` interface (refer to Chapter 30: slide 21-23). *[Graph.java]*

2. Define `AbstractGraph<V>` abstract class that implements `Graph<V>` interface (refer to Chapter 30: slide 21-23). *[AbstractGraph.java]*
   - Define an inner class `Tree` for depth-first search and breadth-first search

3. Define `UnweightedGraph<V>` concrete class that extends `AbstractGraph<V>` abstract class (refer to Chapter 30: slide 21-23). *[UnweightedGraph.java]*

4. Write the test program that builds a graph with 12 cities and their edges. Then, the program prints out all the edges of each city in the graph. Also, the program prints a DFS and a BFS for the graph. *[ProgrammingExerciseP8.java]*

   [Guideline:]

1. *Build a graph with 12 cities and their edges & Print out all the edges of each cities: refer to sample program – Chapter 28 > TestGraph.java*
2. *Print DFS for the graph: refer to sample program – Chapter 28 > TestDFS.java*
3. *Print BFS for the graph: refer to sample program – Chapter 28 > TestBFS.java*