TK1143 Program Design

Graphs

Section A:

1. Describe the following terms with an example:

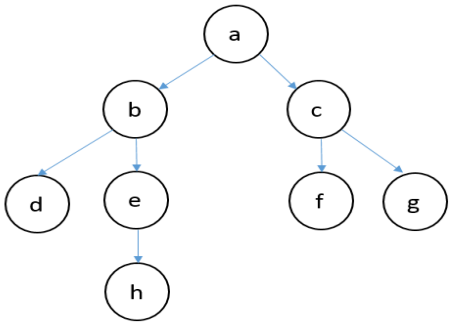
   a) Edges
   b) Vertices/Nodes
   c) Directed Graph @ Digraph
   d) Undirected graph
   e) Connected Graph
   f) Disconnected Graph
   g) Weighted graph
   h) Unweighted graph
   i) Path/cycle
   j) Adjacent
   k) Loop

2. To model a problem with a graph, we need to determine the vertices and edges.

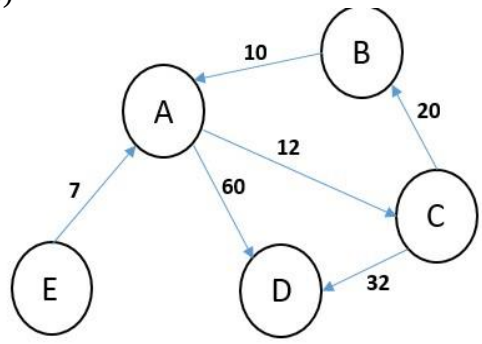
	graphs	vertices	edges
a	Maps		
b	Social Networks		
c	A Chess Game		
d	Telephone Lines		
e	Transportation		

3. Draw the (i) adjacency matrix and (ii) adjacency list representation for the following directed graphs:

a)



b)



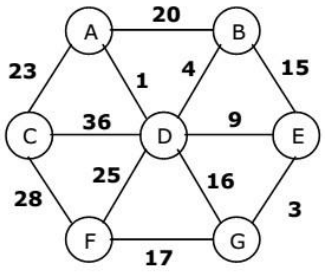
4. Let *G* be an undirected graph whose vertices are the integers 1 through 8, and let the adjacent vertices of each vertex given by the table below:

vertex	adjacent vertices
1	(2, 3, 4)
2	(1, 3, 4)
3	(1, 2, 4)
4	(1, 2, 3, 6)
5	(6, 7, 8)
6	(4, 5, 7)
7	(5, 6, 8)
8	(5, 7)

Assume that, in a traversal of *G*, the adjacent vertices of a given vertex are returned in the same order as they are listed in the table above.

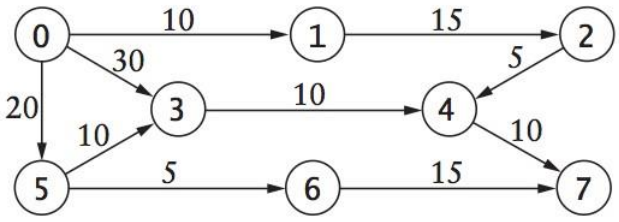
a. Draw *G*.
b. Draw the adjacency matrix of the graph.
c. Draw the adjacency list of the graph.
d. List the nodes of the graph in a DFS traversal starting at vertex 1.
e. List the nodes of the graph in a BFS traversal starting at vertex 1.

5. For the figure shown below, what is the output for depth first spanning tree visiting sequence?

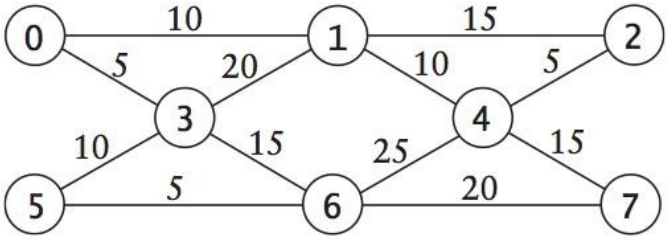


Node	Adjacency List
A	B C D
B	A D E
C	A D F
D	A B C E F G
E	B D G
F	C D G
G	F D E

6. Based on the following graph, find the shortest path from vertex 0 to 4.



7. In the following graph, draw and find the minimum spanning tree.



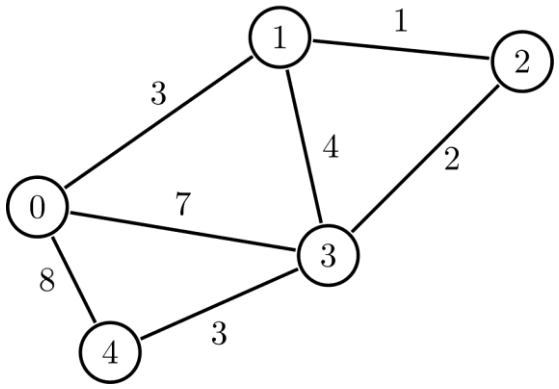
Section B:

1. Consider the following code. (*Refer Slide Note*)

```
1  import java.util.LinkedList;
2
3  public class WeightedGraph {
4      static class Edge {
5          int source;
6          int destination;
7          int weight;
8
9          public Edge(int start, int end, int km) {
10
11
12
13         }
14     }
15
16     static class Graph {
17         int vertices;
18         LinkedList<Edge> [] adjacencylist;
19
20         Graph(int vertices) {
21             this.vertices = vertices;
22             adjacencylist = new LinkedList[vertices];
23             //initialise adjacency lists for all the vertices
24
25
26             }
27         }
28
29         public void addEgde(int source, int destination, int weight) {
30             Edge edge = new Edge(source, destination, weight);
31             adjacencylist[source].addFirst(edge); //for directed graph
32         }
33
34         public void printGraph(){
35             for (int i = 0; i <vertices ; i++) {
36                 LinkedList<Edge> list = adjacencylist[i];
37                 for (int j = 0; j <list.size() ; j++) {
38                     System.out.println("vertex-" + i +
39                     " is connected to " +
40                     list.get(j).destination + " with weight " +
41                     list.get(j).weight);
42                 }
43             }
44         }
45     }
```

```
47      public static void main(String[] args) {
48            int vertices = 6;
49            Graph graph = new Graph(vertices);
50            graph.addEgde(0, 1, 4);
51            graph.addEgde(0, 2, 3);
52            graph.addEgde(1, 3, 2);
53            graph.addEgde(1, 2, 5);
54            graph.addEgde(2, 3, 7);
55            graph.addEgde(3, 4, 2);
56            graph.addEgde(4, 0, 4);
57            graph.addEgde(4, 1, 4);
58            graph.addEgde(4, 5, 6);
59            graph.printGraph();
60        }
61  }
62
```

a. What suppose you initialize in line 9-13.
b. Write the code supposedly have in line 24-27 to initialize adjacency lists for all vertices.
c. Draw the graph based on code line 48-60.
d. Draw the adjacency matrix of the graph.
e. Draw the adjacency list of the graph.
f. List the output of the program.
g. Change code in line 50-60 based on graph below.



2. Consider the following code.

```
1  import java.io.*;
2  import java.util.*;
3
4  class Graph
5  {
6      private int V;
7
8      // Array of lists for Adjacency List Representation
9      private LinkedList<Integer> adj[];
10
11     // Constructor
12     Graph(int v)
13     {
14         V = v;
15         adj = new LinkedList[v];
16         for (int i=0; i<v; ++i)
17             adj[i] = new LinkedList();
18     }
19
20     //Function to add an edge into the graph
21
22
23
24
25
26     // A function used by DFS
27     void DFSUtil(int v,boolean visited[])
28     {
29         // Mark the current node as visited and print it
30         visited[v] = true;
31         System.out.print(v+" ");
32
33         // Recur for all the vertices adjacent to this vertex
34         Iterator<Integer> i = adj[v].listIterator();
35         while (i.hasNext())
36         {
37             int n = i.next();
38             if (!visited[n])
39                 DFSUtil(n, visited);
40         }
41     }
42
43     // The function to do DFS traversal. It uses recursive DFSUtil()
44     void DFS(int v)
45     {
46         // Mark all the vertices as not visited(set as
47         // false by default in java)
48         boolean visited[] = new boolean[V];
49
50         // Call the recursive helper function to print DFS traversal
51         DFSUtil(v, visited);
52     }
53 }
```

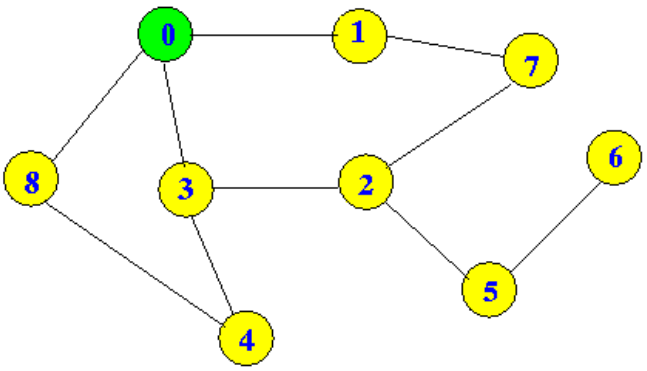
```
public static void main(String args[])
{
    DFS g = new DFS(4);

    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 0);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 1);
    g.addEdge(2, 3);
    g.addEdge(3, 2);
    g.addEdge(3, 3);

    System.out.println("Following is Depth First Traversal "+
                       "(starting from vertex 2)");

                       // to set DFS starting from vertex 2
}
```

a. What the value of V in line 6.
b. Write the code supposedly have in line 21-24 to add an edge into the graph.
c. Write the code to set DFS starting from vertex 2 in main method.
d. Draw the graph based on code given.
e. What the output based on code given.
f. Change code in main method based on graph below which starting from vertex 0.

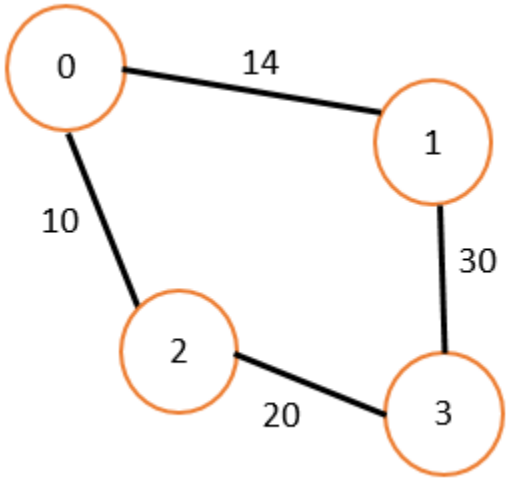


Section C:

Code Zinger: Practice 11.1	
Simple Graph	
Input	Standard Input
Output	Standard Output
Data Structure	Graph (Dijkstra Algorithm)

Problem

Base on the graph below find the shortest path.



Sample Input-Ouput

Input	Output
0 2	The shorted path from 0 to 2 is 10
0 3	The shorted path from 0 to 3 is 30
2 3	The shorted path from 2 to 3 is 20
1 3	The shorted path from 1 to 3 is 30

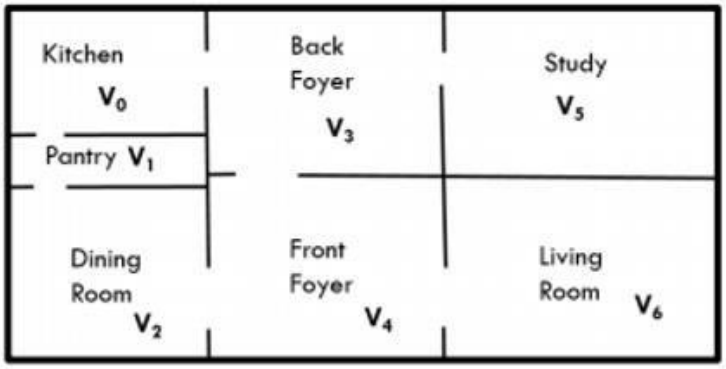
Notes: 1) Dijkstra codes are given.
2) Create an application class to display the output.

Code Zinger: Practice 11.2	
Home To Sell	
Input	Standard Input
Output	Standard Output
Data Structure	Graph (DFS Algorithm)

Problem

The client wants to explore the house's area that you want to sell as figure below. Your client is allowed to decide where area they want to start. Your task is to find the areas in the house to the client.

*Notes:Implement the DFS Algorithm.



Input

4

*Notes: The number of the vertex that which part of the home to start.

Sample Input-Output

Input	Output
4	Front Foyer -> Dining Room -> Pantry -> Kitchen -> Back Foyer -> Study Room -> Living Room ->
0	Kitchen -> Pantry -> Dining Room -> Front Foyer -> Back Foyer -> Study Room -> Living Room ->

Notes: 1) Modified the DFS codes
- to set the area name using array.
- the array of lists for adjacency list representation
- the method addEdge ( ) to add edge into the graph

2) Create an application class
- to add each of the vertex
- input the number of the vertex that which part of the home to start.

Code Zinger: Assignment 11.3	
Malacca Map	
Input	Standard Input
Output	Standard Output
Data Structure	Graph (Dijkstra Algorithm)

Problem

Figure 1 is a map of Malacca. The circle location is a small city in Malacca. Figure 2 is a map of Malacca in graphs forms with distances for each small city. Find the shortest path between main small cities (refer Table 1).

*Notes: Implement the Dijkstra Algorithm.

Figure 1

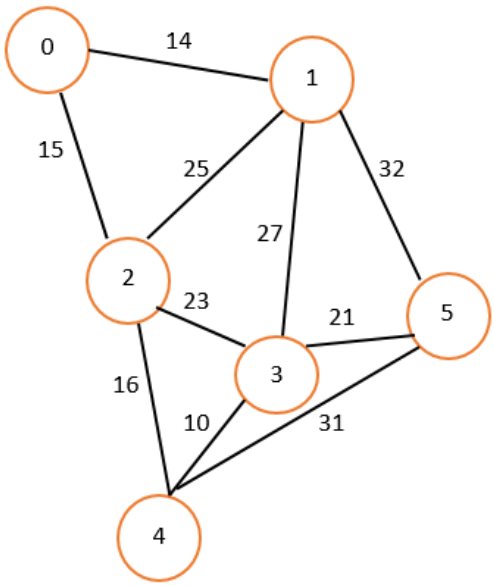


Figure 2

Table 1

Node	District
0	Masjid Tanah
1	Alor Gajah
2	Tangga Batu
3	Bukit Katil
4	Kota Melaka
5	Jasin

Input

1
3

Notes: First line is the start of destination
The second line is the destination

Sample Input-Ouput

Input	Output
1 3	The Shorted Path from Alor Gajah to Bukit Katil is 27 km
4 5	The Shorted Path from Kota Melaka to Jasin is 31 km

Notes: 1) Dijkstra codes are given.
2) Create an application class to display the output.

Code Zinger: Assignment 11.4	
Penang Island	
Input	Standard Input
Output	Standard Output
Data Structure	Graph Prim's Minimum Spanning Tree (MST) Algorithm

Problem

There are eight exciting destinations around Penang Island. The state wants to build Light Rapid Transit (LRT) to connect them so that each fascinating destination can be reached from any other one via one or more LRT. The cost of constructing an LRT is proportional to its length. The distances between pairs of destinations are given in the following table. Find which MRT to build to minimize the total construction cost.

*Notes: Implement the *Prim's Minimum Spanning Tree (MST)* algorithm.

Penang	Airport	B.Bendera	Bt.Ferrenghi	Tg.Tokong	U.S.M	Muka Head	Komtar	B.Jambul
Airport	-	25	31	22	9	37	24	6
B.Bendera	-	-	11	9	18	25	11	20
Bt. Ferenggi	-	-	-	10	23	6	16	27
Tg.Tokong	-	-	-	-	14	17	6	17
U.S.M	-	-	-	-	-	30	11	5
Muka Head	-	-	-	-	-	-	22	33
Komtar	-	-	-	-	-	-	-	14
B.Jambul	-	-	-	-	-	-	-	-

Sample Input-Output

Input	Output
-	Destination[Length] Tg.Tokong - B.Bendera[9km] Tg.Tokong - Bt.Ferrenghi[10km] Komtar - Tg.Tokong[6km] B.Jambul - U.S.M[5km] Bt.Ferrenghi - Muka Head[6km] U.S.M - Komtar[11km] Airport - B.Jambul[6km]

Notes: 1) Modified the MST codes
- to set the destination name using array
- the method printMST( ) to display the destination and length

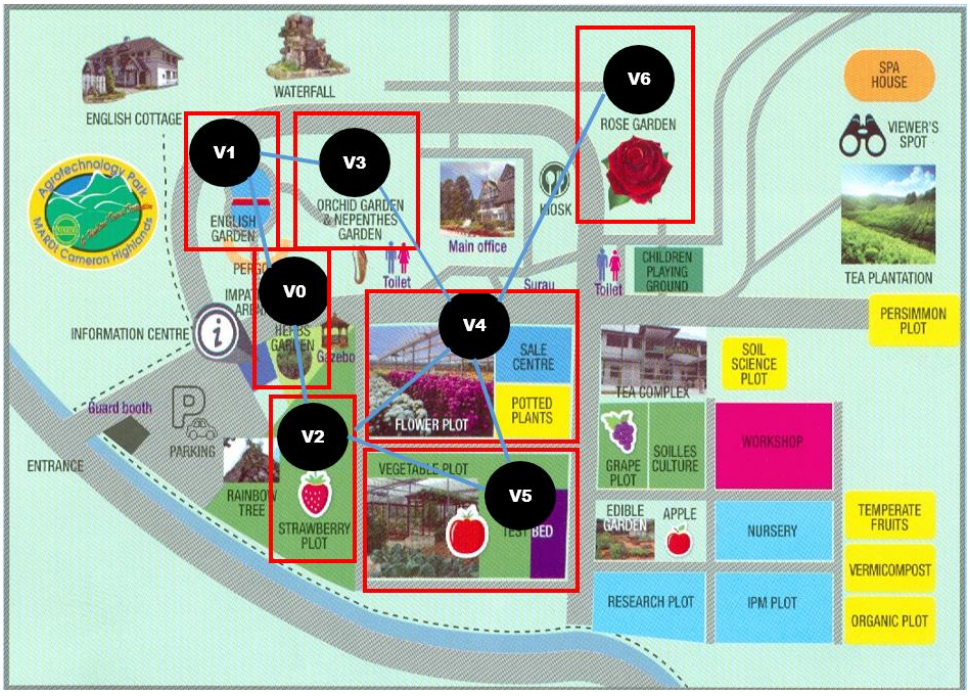
2) Create an application class to set adjacency matric based on the table.

Code Zinger: Assignment 11.5	
Mardi Cameron Highlands	
Input	Standard Input
Output	Standard Output
Data Structure	Graph (BFS Algorithm)

Problem

The Agro-Technology Park MARDI Cameron Highlands is located within the MARDI Cameron Highland's Stations and was officiated by His Royal Highness DYMM, The Sultan of Pahang on 14th of June, 2003. Since then, about 65,000 visitors, visit the park each year. Among the major attractions are the Herbs Garden, Strawberry Plot, English Garden, Orchid & Nepenthes Garden, Vegetable Plot, Flower Plot and Rose Garden. Your task is to find the attractions place in Agro-Technology Park MARDI to the visitors.

*Notes: Implement the BFS Algorithm.



Input
0

Notes: The number of the vertex that which part of the home to start.

Sample Input-Output

Input	Output
0	Herbs Garden -> Strawberry Plot -> English Garden -> Orchid & Nepenthes Garden -> Vegetable Plot -> Flower Plot -> Rose Garden ->
2	English Garden -> Herbs Garden -> Vegetable Plot -> Flower Plot -> Strawberry Plot -> Orchid & Nepenthes Garden -> Rose Garden ->

Notes: 1) Modified the BFS codes
- to set the area name using array.
- the method addEdge ( ) to add edge into the graph

2) Create an application class
- to add each of the vertex into graph
- input the number of the vertex that which part of the home to start.