

## **CptS 223 Homework #4 - Graphs**

Please complete the homework problems on the following page using a separate piece of paper. Note that this is an individual assignment and all work must be your own. Be sure to show your work when appropriate.

1. [13] Define these terms as they relate to graph and graph algorithms:

Use mathematical terms where appropriate.

Graph

A Graph is a data structure, which consists of a set of vertices

$V$  and a set of edges  $E$ . A graph is represented by  $G = (V, E)$ .

Where  $V$  is a set vertices and  $E$  is a set of edges.

---

Vertice

Vertex" is a synonym for a node of a graph, i.e., one of the points on which the graph is defined and which may be connected by graph edges.

---

Edge

A line joining two unordered pair of nodes are said to form an edge.

---

## UndirectedGraph

A graph in which edges have no orientation, which is normally represented with a line. Edges can be traversed in either direction.

---

## DirectedGraph

A graph in which edges have orientation and which is normally shown by an arrow.

---

## Path

A path is sequence of vertices which is connected by edges.

A length of path is given by number of edges.

---

## Loop

Loop is an edge with both ends has same vertex.

---

## Cycle

Cycle is defined as closed walks which do not repeat edges or vertices except for the starting and ending vertex.

---

## Acyclic

Acyclic graph is a graph with no cycle.

---

## Connected

A graph is connected when there is a path between every pair of vertices. In a connected graph, there are no unreachable vertices.

---

## Sparse

A graph in which the number of edges is much less than the

possible number of edges.

---

Weight

A weighted graph is a graph in which each edge is given a numerical weight.

---

2. [4] Under what circumstances would we want to use an adjacency matrix instead of an adjacency list to store our graph?

We use an adjacency matrix in undirected graph and we can use them to represent in weighted graph.

we use adjacency matrices instead of an adjacency list if we want to check whether a specific edge between two vertices belongs to the graph because it takes only an  $O(1)$  lookup often. And also we can use adjacency matrix if you need to do insertions and deletions of edges.

3. [6] Name three problems or situations where a graph would be a good data structure to use:

Finding shortest route from one location to another in Google map.

Finding least traffic route.

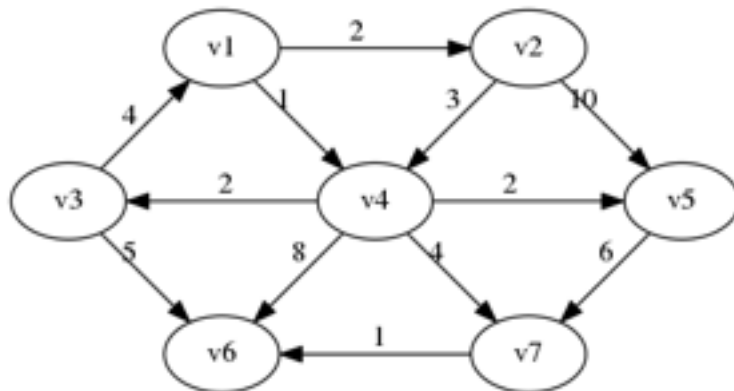
course prerequisite

Graph is used in Networking for internet , telephone

Graph is used in social networking such as LinkedIn and Facebook.

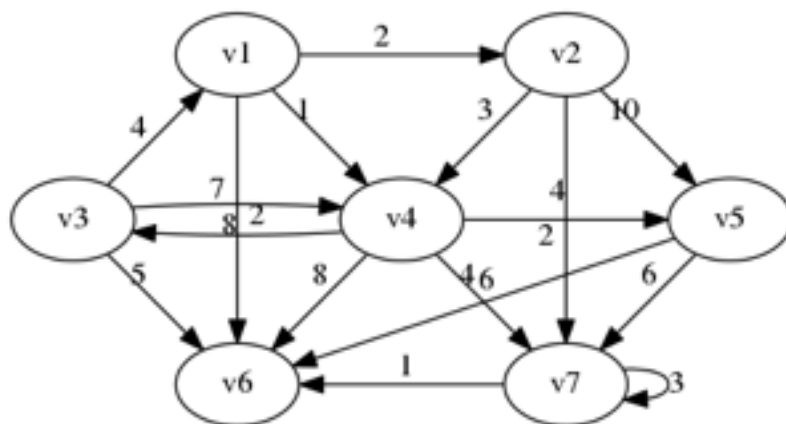
Airport connections.

4. [4] What kind of graph is this?



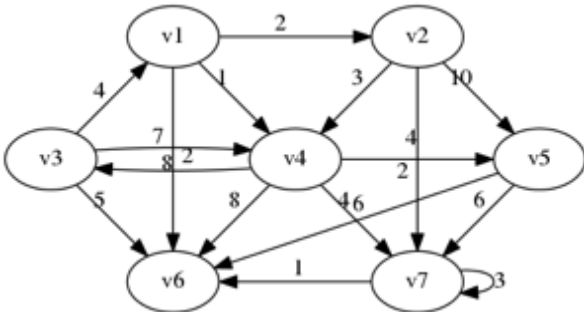
It is a directed cyclic graph. It is directed because there is arrow from one node to other. it is cyclic because there exists a cycle of v1-v2-v4-v3-v1.

5. [4] Identify the loop in this graph:

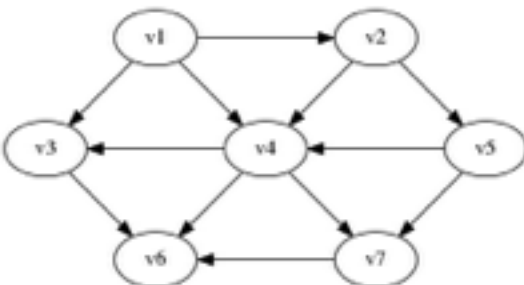
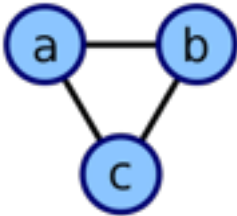
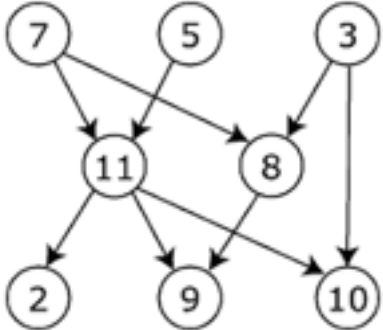


There are three loops in this graph v1-v4-v3-v1 and v3-v4-v3 and v7-v7

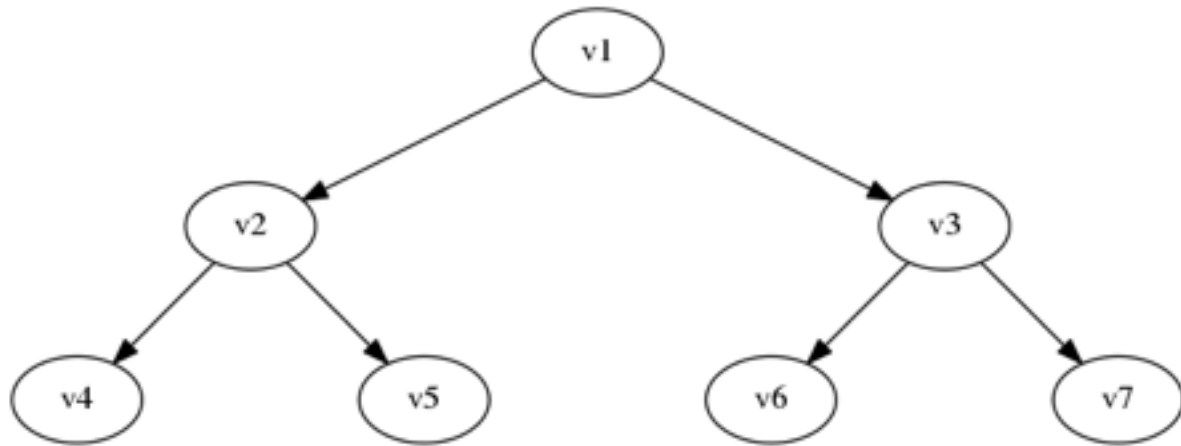
6. [4] How many vertices and edges are in this graph:

	<p>Vertices ____7____</p> <p>Edges ____17____</p>
---	---

7. [6] Are these cyclic or acyclic graphs?

	<p>Cyclic?</p> <p>No, It is Not a cyclic</p>
	<p>Cyclic?</p> <p>Yes, it is a cyclic</p>
	<p>Cyclic?</p> <p>No, it is not a cyclic</p>

8. [5] A tree is a particular kind of graph. What kind of graph is that?



This tree is a directed acyclic connected graph

9. [4] What is the difference between a breadth-first search and a depth first search?

Breadth First Search is a node based technique for finding a shortest path in graph. It uses a Queue data structure. In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue. It is slower than DFS.

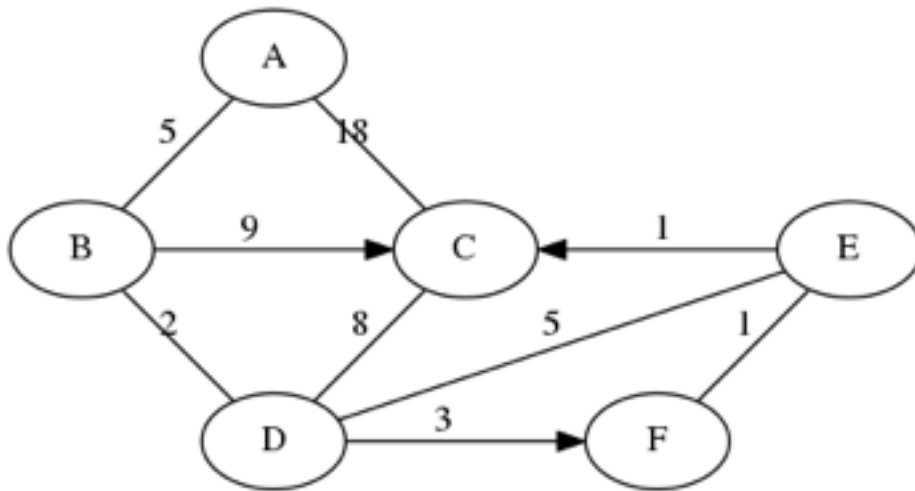
Depth First Search is an edge based technique. It uses the Stack data structure, performs two stages, first visited vertices are pushed into stack and second if there are no vertices then visited vertices are popped.



**10. [10] Dijkstra's Algorithm.** Use Dijkstra's Algorithm to determine the shortest path starting at A. Note that edges without heads are bi-directional. To save time, you do not have to add items to the "priority queue" column after it has been discovered (listed in the "distance" column). Use the table below to show your work.

What's the shortest route (by weight) from A to C?

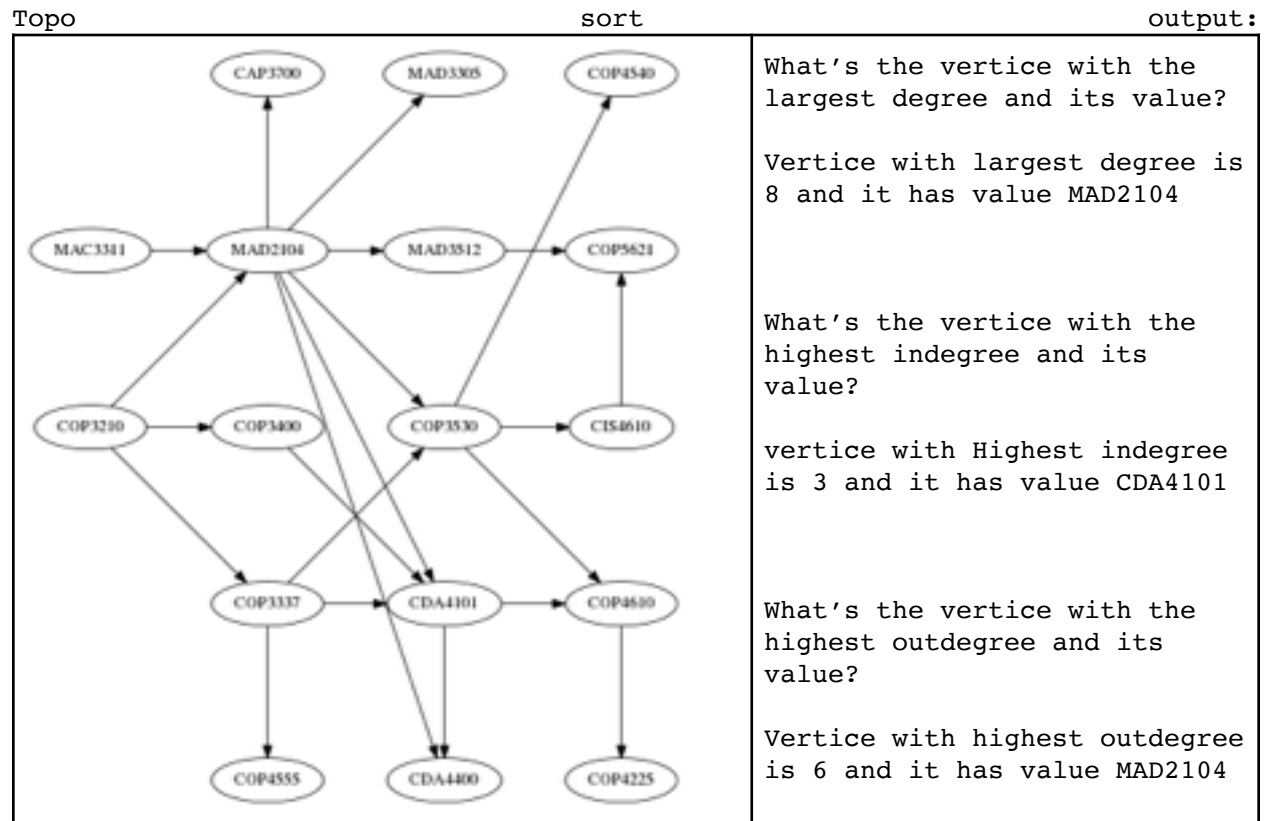
---



The distance from A to C is A-B-D-F-E-C and the shortest distance is 12

Node: Distance	A	B	C	D	E	F
A	0	5A	18	INF	INF	INF
B		5A	14B	7B	INF	INF
C			14B	7B	12D	10D
D			14B		11F	10D
E			12E		11F	
F			12E			

11. [10] **Topo sort.** Show the final output of running Topo Sort on this graph:



MAC3311 COP3210

MAD2104 COP3337 COP3400 CAP3700 MAD3305

CDA4400 CDA4101 COP3530 MAD3512 COP4555

COP4225 COP4610 CIS4610 COP5621 COP4540