

Data structure - Graph

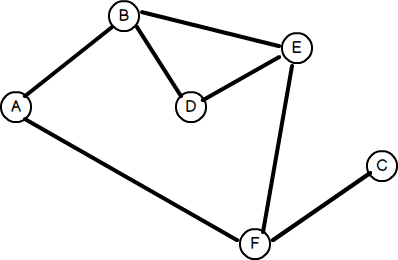
In this session, we will learn: What are graphs?

Representation of graphs.

Implement adjacency list representation of a graph.

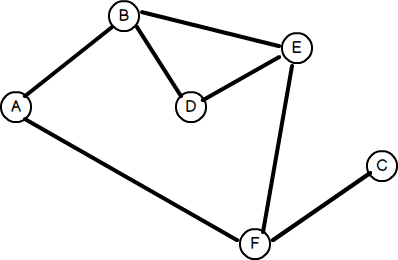
A graph can be defined as collection of finite number of vertices and edges.

Graph is a non linear data structure.



A vertex, a.k.a node in a graph.

It represents information such as name or identifier of a node in a graph.



An edge is a line connecting two adjacent vertices.

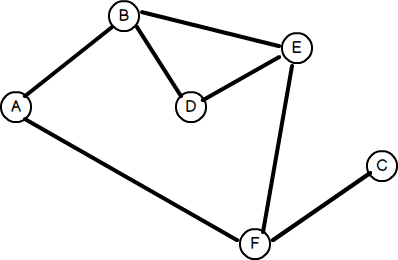
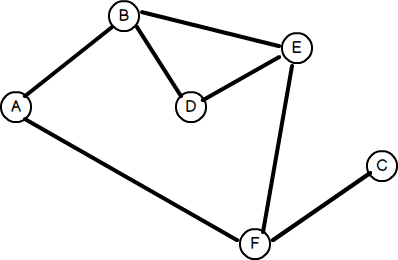


Figure:G

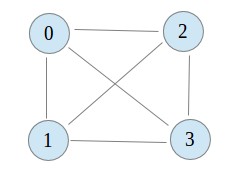


*G* = (*V, E*)

*V* (*G*) = *{A, B, C, D, E, F}*

*E*(*G*) = *{*(*A, B*)*,* (*B, D*)*,* (*B, E*)*,* (*A, F* )*,* (*F, E*)*,* (*C, F* )*}*

Figure:G



In a given graph *G*, *∃*(*n −* 1) adjacent vertices

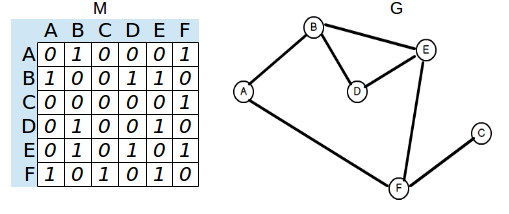
*∀Gv*, where *n* in number of vertices in *G*.

*G* can be represented as square Matrix *M*

of values 0*s* and 1*s*.

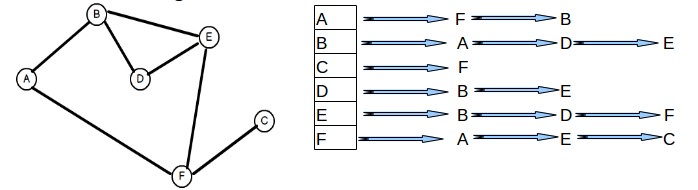
*Mij* = 1*, ∃ E*(*vi ,vj* ), else *Mij* = 0.

*Mi* &*Mj* represents *V* (*G*).



Graph *G* can be represented as array of lists where *Ai* is list of adjacent vertices of *Vi* from *G*(*V* ).

Figure:Adjacency list representation of a graph.



Implementing functions



1

*node \*newNode(int vertex)*, returns a new node with

**vertex** in it.

*adjacencyList \*createGraph(int n)*, returns an array which represents *G* with *n* vertices.

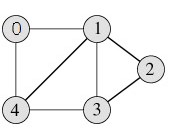


2

3 *void displayGraph(int n)*, prints *G* of *n* vertices.

4 *void addAdjacentVertex(int vertex, int adjVertex)*, adds

**adjVertex** as adjacent vertex to **vertex**.

Considering the following graph.

Graph definition.

Graph representation as a data structure. Implementing adjacency list for a graph.

