Graphs

Prepared by Mahdi Ghamkhari

Graphs

A graph is a non-linear data structure made up of

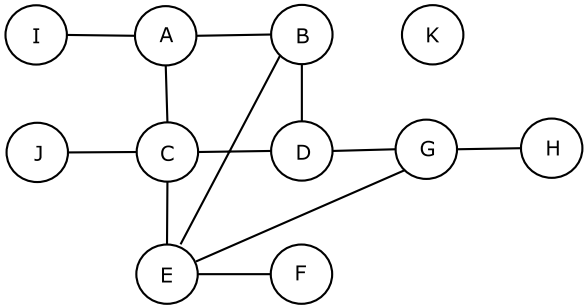
- vertices
- Edges

The vertices are connected by the edges

Graphs

How many Vertexes? 11

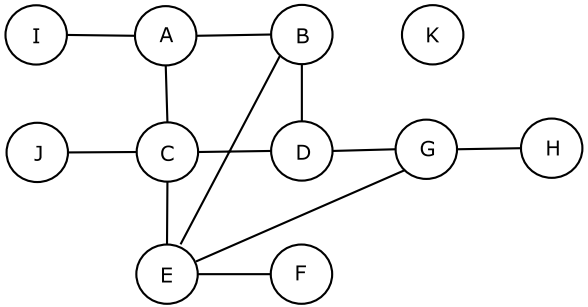
How many edges? 12



Graph terminology

- Connected

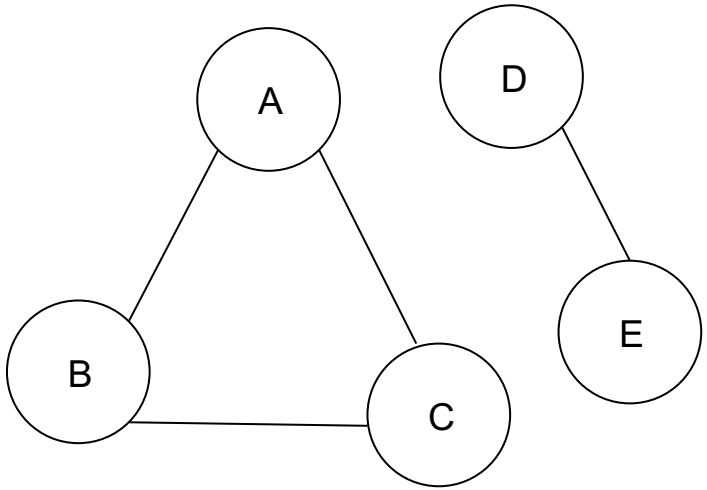
A graph is connected if there is a path between any two vertexes



- Cyclic

A graph is connected if there is at least one vertex that has a path to itself

A graph that is cyclic, but not connected?

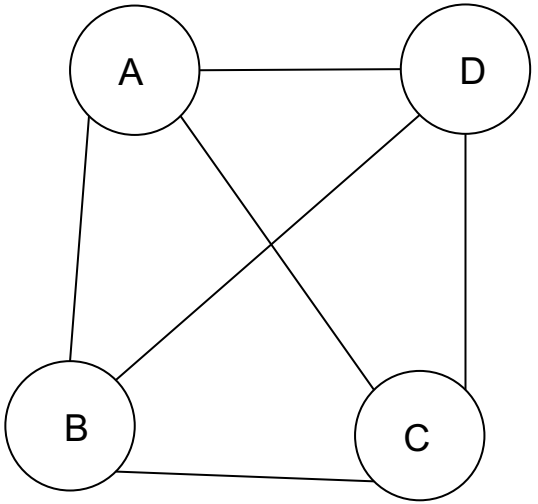


Complete graphs:

In a complete graph, there is an edge betweenn every two vertexes

The number of edges in a complete graph with n vertexes?

n(n-1)/2

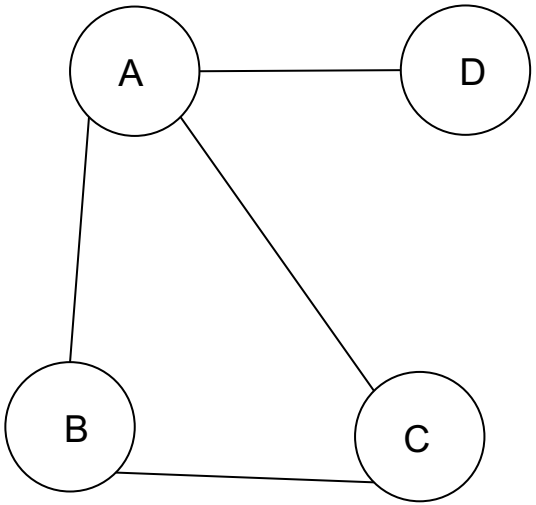


Graph implementation

We can implement a graph using an adjacency matrix.

Graph implementation

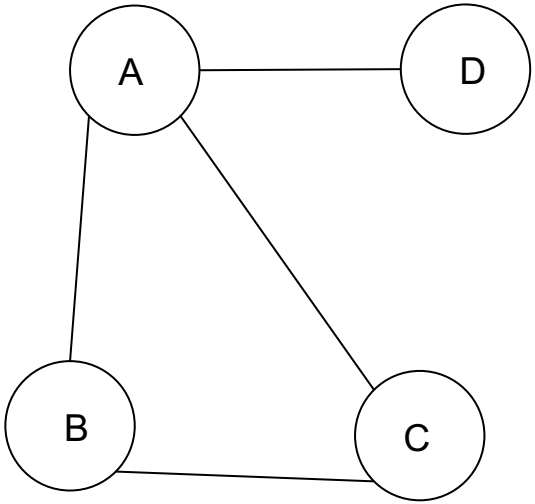
A and B are neighbor/adjacent
A and C are neighbor/adjacent
A and D are neighbor/adjacent
B and C are neighbor/adjacent



Graph implementation

A and B are neighbor/adjacent
A and C are neighbor/adjacent
A and D are neighbor/adjacent
B and C are neighbor/adjacent

	A	B	C	D
A	**false**	**True**	**True**	**True**
B	**True**	**false**	**True**	**false**
C	**True**	**True**	**false**	**false**
D	**True**	**false**	**false**	**false**



Adjacency matrix

- The adjacency matrix is symmetrical.
- Adding an edge has O(1) time complexity, as we only need to edit two entries of the adjacency matrix.
- Adding a vertex has O(n^2) time complexity, as we need to create a new adjacency matrix and copy the previous one into the top-left corner of the new one.

Traversing a graph:

Starting from a starting vertex, and reaching to all other vertexes that are reachable, visiting each vertex once.

Traversing a graph:

Depth first

Breadth first

Breadth-first traversal

Visit a starting vertex, then
Visit the neighbors of the starting vertex, then
Visit the neighbors of the neighbors of the starting vertex

As we visit vertexes, we keep track of which ones we have visited so that we don't visit a vertex twice.

Algorithm for breadth-first traversal

enQueue the starting vertex in a queue

While (Queue is not empty)
    deQueue a vertex from the Queue.
    Visit the vertex
    enQueue unvisited neighbors of the vertex.

Depth-first traversal:

Visit a starting vertex
Keep moving away from the starting vertex, visit new vertexes until reaching to a dead-end.
Move back to an unvisited vertex that is closest to the starting vertex, and again keep moving away.

An algorithm for depth-first traversal:

Visit the starting vertex, and push it on a stack

While(stack is not empty)

- Withdraw a vertex from the stack,
- If the vertex has no unvisited neighbor, continue with the next iteration
- Otherwise, visit one of the unvisited neighbors
- Push the vertex on the stack again
- Push the visited neighbor on the stack