

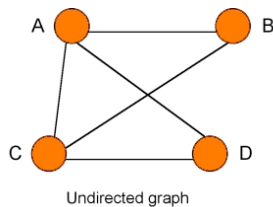
# GRAPHS

Discrete structures consisting of vertices and edges that connect these vertices.

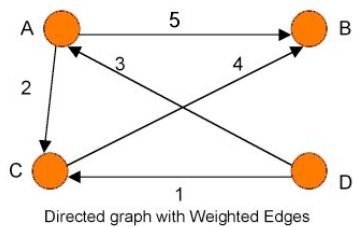
Divya Bansal  
Research Assistant, Computer Science

## BASICS

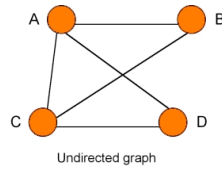
- **UNDIRECTED GRAPHS:** we can move in both directions between vertices.



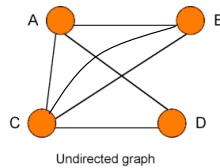
- **DIRECTED GRAPHS:** direction of any given edge is defined and weights can be assigned to the edges.



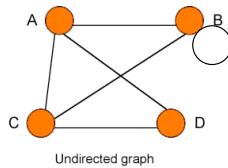
- **SIMPLE GRAPH:**  $G=(V,E)$  consists of  $V$ , a non empty set of vertices, and  $E$ , a set of unordered pairs of distinct elements of  $V$  called edges.



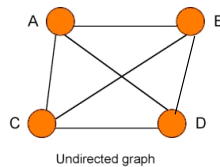
- **MULTIGRAPH:** It is a simple graph, but multiple edges between vertices are allowed. So every multigraph is a simple graph but every simple graph is not a multigraph.



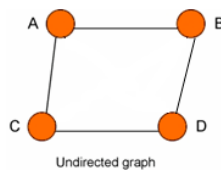
- **LOOPS:** These are edges from a vertex to itself. Not allowed in multigraph.
- **PSEUDOGRAPH:** A multigraph, but loops allowed.



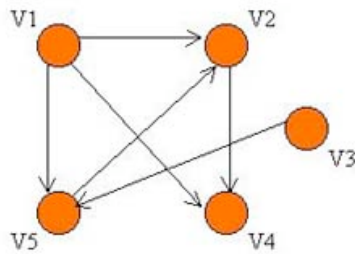
- **COMPLETE GRAPH:** it's a simple graph that contains exactly one edge between each pair of distinct vertices.



- **CYCLES:** The cycle  $C_n$ ,  $n \geq 3$ , consists of  $n$  vertices  $v_1, v_2, \dots, v_n$  and edges  $\{v_1, v_2\}$ ,  $\{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}$ , and  $\{v_n, v_1\}$ .



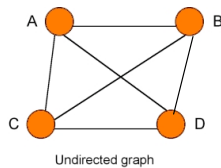
- **ADJACENCY MATRICES:** Graphs can also be represented in the form of matrices.
  - **Advantage** of matrix representation is that the calculation of paths and cycles can easily be performed using well known operations of matrices.
  - **Disadvantage** is that this form of representation takes away from the visual aspect of graphs. It would be difficult to illustrate in a matrix, properties that are easily illustrated graphically.



	v1	v2	v3	v4	v5
v1	0	1	0	1	1
v2	0	0	0	1	0
v3	0	0	0	0	1
v4	0	0	0	0	0
v5	0	1	0	0	0

- In case of **undirected graphs** there is value 1 in both the entries i.e. from A to B and from B to A
- In case of multigraphs and pseudographs (undirected) it is **no more a zero-one matrix**. Instead it is filled with the number of paths between the vertices.
- Adjacency matrix for undirected graphs are **symmetric**.

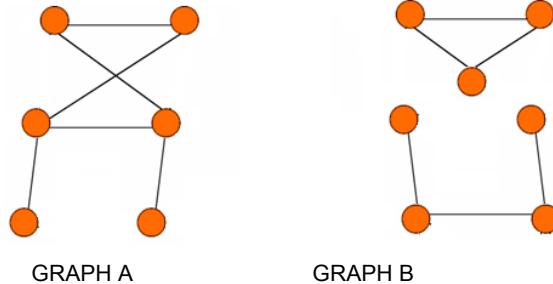
- **PATH:** A path through a graph is a traversal of consecutive vertices along a sequence of edges.
  - the vertex at the end of one edge in the sequence must also be the vertex at the beginning of the next edge in the sequence.
  - The vertices that begin and end the path are termed the **initial vertex** and **terminal vertex**, respectively.
  - **Length** of the path is the number of edges that are traversed along the path.
- **CIRCUIT:** The path is a circuit/cycle if it begins and ends at the same vertex and the length is greater than zero.



Here ACBD is a path.  
And ACBDA is a circuit.

- **CONNECTEDNESS IN UNDIRECTED GRAPHS:**

- An undirected graph is considered to be **connected** if a path exists between all pairs of vertices thus making each of the vertices in a pair reachable from the other.
- A graph that is not connected is the union of two or more **connected subgraphs**, each pair of which has no vertex in common. These disjoint connected subgraphs are called the connected components of the graphs.

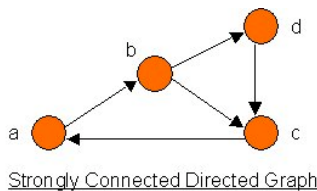


Here Graph A is connected but Graph B is not connected. But the subgraphs of Graph B are connected So it is a connected subgraph.

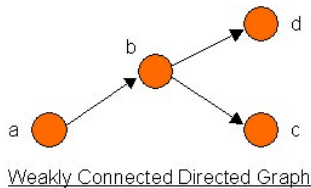
- Sometimes removing a vertex  $v$  and all of the edges incident to  $v$  produces a subgraph with more connected components that the original graph. The vertex is called a **cut vertex** or an **articulation point**.

- **CONNECTEDNESS IN DIRECTED GRAPHS:**

- A directed graph  $G = (V,E)$  is **strongly connected** if there are paths from both  $u$  to  $v$  and  $v$  to  $u$  for all distinct  $u, v$  belongs to  $V$ .



- $G$  is **weakly connected** if there is a path between and two distinct vertices in the underlying undirected graph.

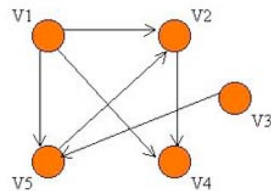


- The maximal strongly connected subgraphs of  $G$  are **strongly connected components**.

- **COUNTING PATHS BETWEEN VERTICES:**

- As shown in the previous example, the existence of an edge between two vertices  $v_i$  and  $v_j$  is shown by an entry of 1 in the  $i$ th row and  $j$ th column of the adjacency matrix. This entry represents a path of length 1 from  $v_i$  to  $v_j$ .
- To compute a path of length 2, the matrix of length 1 must be multiplied by itself, and the product matrix is the matrix representation of path of length 2.

- *Original Graph G:*



- *Matrix representation of path of length 2*

	<b>v1</b>	<b>v2</b>	<b>v3</b>	<b>v4</b>	<b>v5</b>
<b>v1</b>	0	1	0	1	0
<b>v2</b>	0	0	0	0	0
<b>v3</b>	0	1	0	0	0
<b>v4</b>	0	0	0	0	0
<b>v5</b>	0	0	0	1	0

- *The above matrix indicates that we can go from vertex v1 to vertex v2, or from vertex v1 to vertex v4 in two moves. In fact, if we examine the graph, we can see that this can be done by going through vertex v5 and through vertex v2 respectively. We can also reach vertex v2 from v3, and vertex v4 from v5, all in two moves.*
- In general, to generate the matrix of path of length  $n$ , take the matrix of path of length  $n-1$ , and multiply it with the matrix of path of length 1.

- **SHORTEST PATH PROBLEMS:**

- Many problems can be modeled with the weights assigned to their edges.
- Example:
  - Airline system can be modeled for the following cases:
    - Distance
    - Flight time
    - Fares

- **SHORTEST PATH ALGORITHM:**

Procedure Dijkstra ( $G = (V,E)$  with  $w: V \times V \rightarrow \mathbf{R}^+$ .  $G$  is a weighted connected simple graph,  $a, z \in V$ : initial and terminal vertices )

```
for i := 1 to n
    L(i) := ∞
L(a) := 0
S := ∅
while z ∉ S
    u := a vertex not in S with L(u) minimal
    S := S ∪ {u}
    for all v ∈ V such that v ∉ S
        if L(u) + w(u,v) < L(v) then L(v) := L(u,v) + w(u,v)
{ L(z) = length of shortest path from a to z. }
```

- **TRAVELLING SALESMAN PROBLEM:**

- Given a number of cities and the costs of traveling from any city to any other city, what is the cheapest round-trip route that visits each city exactly once and then returns to the starting city?
- So, according to graphs theory, it is asking for the circuit of minimum total weight in a weighted, complete, undirected graph that visits each vertex exactly once and return to its starting points.
- If there are  $n$  vertices in a graph and once a starting point is chosen, then there are  $(n-1)!$  Different circuits, out of which half are the circuits in reverse order. So we need to consider only  $(n-1)!/2$  circuits.