**GRAPHS**

**Types of Graphs**

* **Simple graph:** Maximum 1 edge connecting pair of vertices.
* **Multigraph:** Opposite of simple graph.
* **Pseudo graph:** Multigraph with loops.
* **Mixed graph:** Directed + Undirected
* **Regular graph:** Every vertex with same degree.
* **Wheel graph:** All vertices having degree 3 except the one in middle.
* **N-cube:** Bit length & dimensional changes.

**Bipartite Graphs**

* **Bipartite graph:** 2 sets of graph (v1 & v2). An vertex from v1 connect to v2 only.
* **Complementary graphs:** Non-bipartite graphs.

**Adjacency Matrix**

* Make matrix with column & rows with **vertices name**.
* Write 1 if pair exists.
* Its different for directed and undirected graphs.

**Indices Matrix**

* Make matrix with **vertices as row** & **column as edges**.
* It is also different for directed and undirected graphs.

**Isomorphism**

Two graphs satisfying these are isomorphic:-

* Same number of vertices.
* Same number of edges.
* Equal degree of each vertex.
* Correspondence of edges are same.

**Graphical Terms**

* **Walk:** Random traversing vertex to vertex.
* **Trail:** Edge can’t be repeated when traversing.
* **Circuit:** Trail + Start & end vertex are same
* **Path:** Neither vertex nor edge can be repeated.
* **Cycle:** Path + Start & end vertex are same
* **Trivial walk:** Traversing in isolated graph (non-sense).
* **K-cycle:** K is length.
* **Connected graph:** Path in undirected graph.
* **Connected component:** Subgraph of connected graph.
* **Cut vertex:** Removing a vertex and its related edges.
* **Strongly connected:** Bidirectional path.
* **Weakly connected:** Non-bidirectional path.

**Eulerian Graph Terms**

* Its all about **edges**.
* **Eulerian circuit:** A circuit containing all edges of a graph.
* **Eulerian path:** Path containing all edges.
* **Eulerian graph:** Connected Eulerian circuit.

**Hamiltonian Graph Terms**

* Its all about **vertices**.

**Special Algorithms**

* **Dijkstra’s Algorithm:** Fix and find connective paths.
* **Chromatic colouring:** **Least** number of colours required for a graph.
* **Kruskal’s Algorithm:-**
  + Keep choosing edge with minimum edge.
  + Don’t end up making a circuit.
  + Stop after (n-1) edges. **[n = number of edges]**
* **Prim’s Algorithm:-**
  + Select any vertex, and then its smallest edge.
  + Choose next smallest edge from there.
  + Don’t end up making a circuit.
  + Continue until all vertices included.

**Unique to Software Engineering**

* **Dense graph:** Edges are more than vertices.
* **Sparse graph:** Edges are less than vertices.

**Breadth First Search (BFS)**

* **Contains two queues:**
  + Visited
  + Queue
* Visit each node one by one.
* Push each visited and neighbor node to Visited.
* Push 0 to Queue at first.
* Then push unvisited neighbors in Queue.
* Now **repeat** it by **dequeuing** first element at each iteration.