Chapter 12

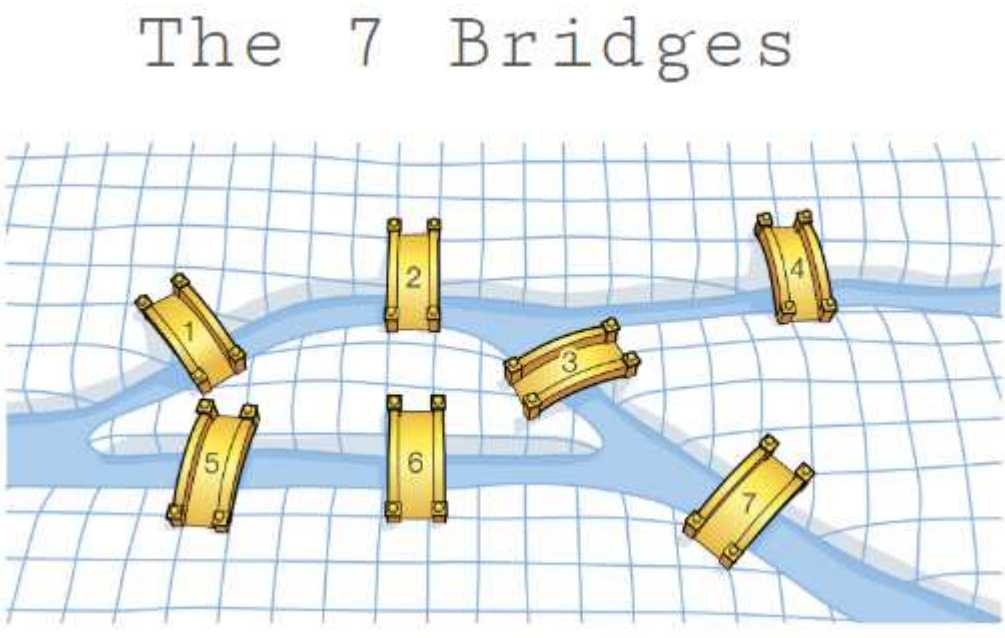
Introduction to Graph Theory

R. 12. 1. (***Historical Background***)

One of the most famous cities in mathematics is the medieval German city of Königsberg, which lay on both sides of the Pregel river.

In the center of the river, there lay 2 islands that were connected to each other and the river banks by *7 bridges.*

Something like this:



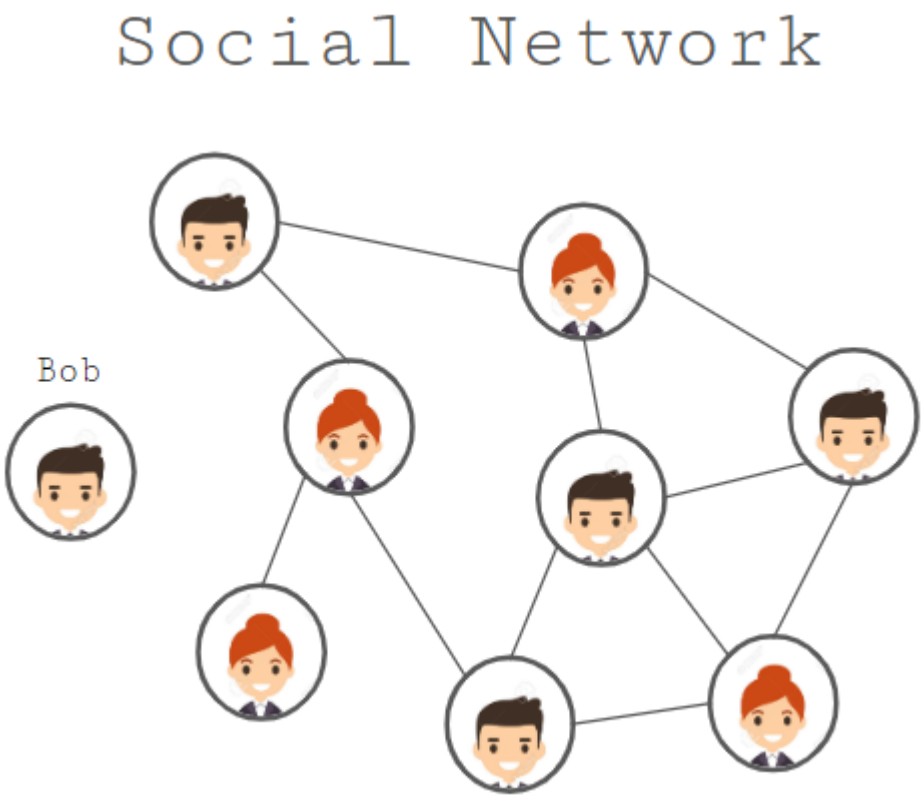
There was a mathematician, Carl Gottlieb Ehler, who later become the mayor of a nearby town, that was absolutely obsessed with finding out a solution for this problem:

Is there a route that would allow someone to cross all 7 bridges without crossing any of them more than once?

Don't try to answer the question. There is no solution to this problem!
But, attempting to *explain why it can't be solved* resulted in the invention of a new field of mathematics by fellow mathematician Leonard Euler. It was a new kind of geometry, termed as *'the geometry of position'*. Today, this is known as the field of graph theory.

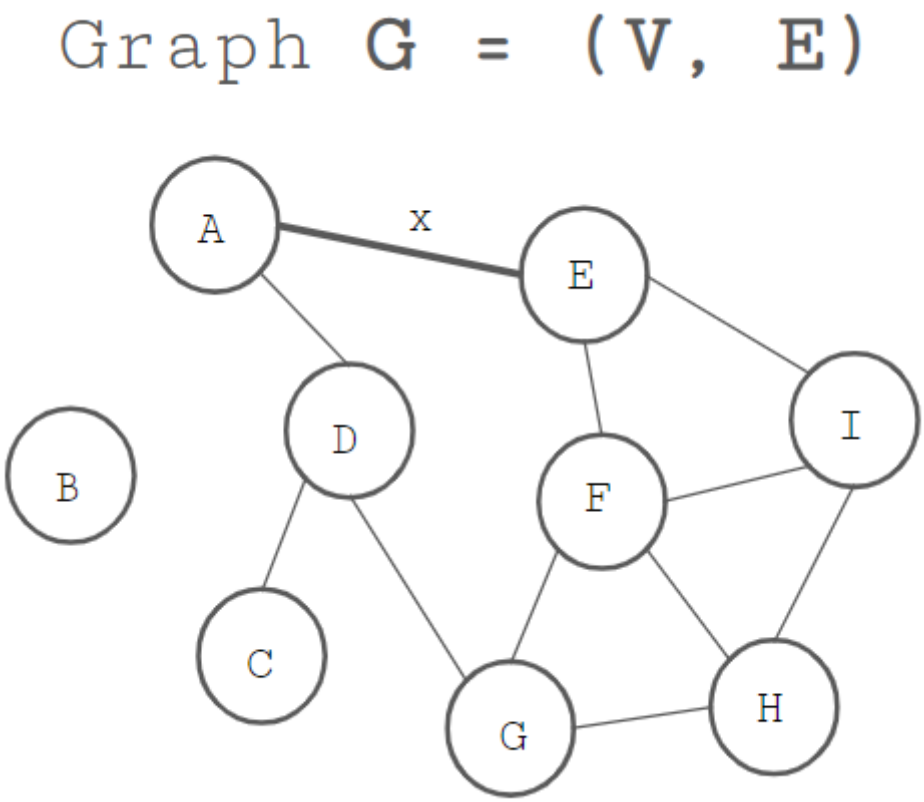
Ex. 12. 1.
Let's take a group of individuals that are connected to some social media platforms, like LinkedIn. If we map out all the connections between these people, it could look something like this.
Each person is a vertex (or node? of the graph. If they are connected on LinkedIn, then there's a line connecting them.

If we take the graph above and 'abstract it', it looks like this:



That's the essence of a graph — a collection of vertices, V, and edges (the lines connecting the people in the social network), E.

If we take the graph above and 'abstract it', it looks lik



D. 12. 1. **(*Some key terms*):**

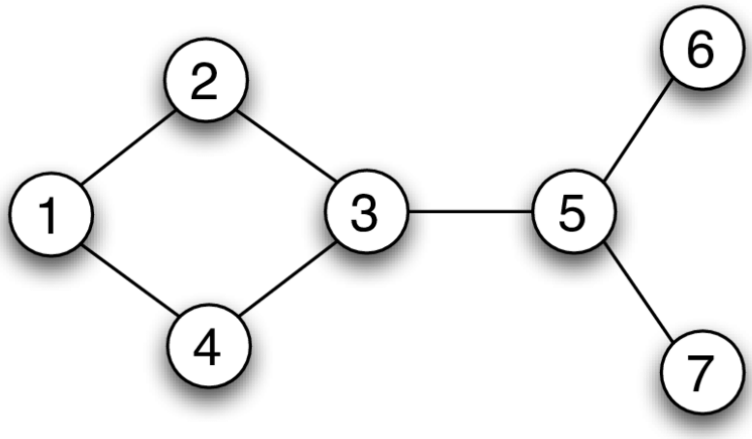
- **Incident**: x is incident to A and E. Any edge is incident to 2 vertices.
- **Adjacent**: G is adjacent to D, F, and H because there is some edge going from G to all these other vertices. Adjacent vertices are connected by an edge.
- **Isolated**: B is isolated because it's not connected to any other vertices (just like how poor Bob doesn't have any connections on LinkedIn…)

- **Degree**: A has a degree of 2, B has a degree of 0, and F has a degree of 4.

  Very simple, the degree is just the number of connections a vertex has.

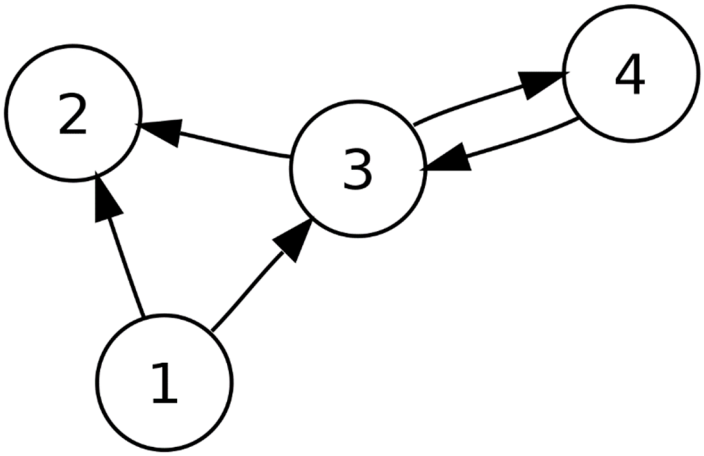
D. 12. 2. (*Types of Graphs*):

- **Undirected Graphs**
  As the name shows, there won't be any specified directions between nodes. So an edge from node A to B would be **identical** to the edge from B to A.



  In the above graph, each node could represent different cities and the edges show the bidirectional roads.

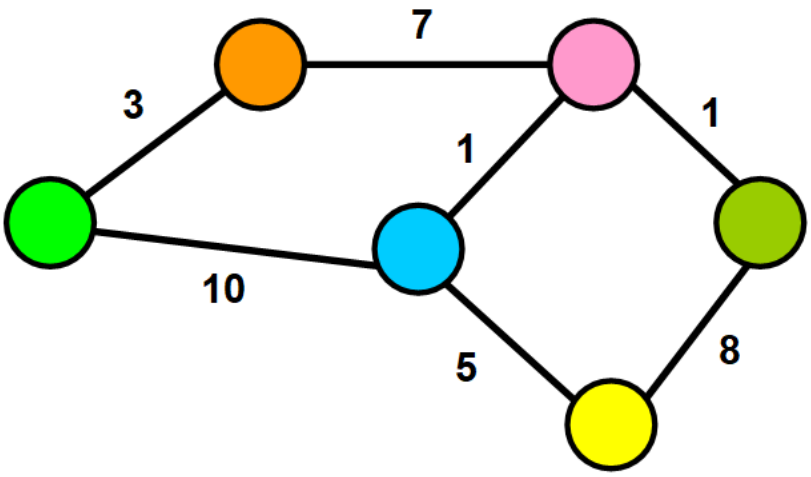
- **Directed Graphs (DiGraphs)**
  Unlike undirected graphs, directed graphs have orientation or **direction** among different nodes. That means if you have an edge from node A to B, you can move only from A to B.



Like the previous example, if we consider nodes as cities, we have a direction from city 1 to 2. That means, you can drive from city 1 to 2 but not back to city 1, because there is no direction back to city 1 from 2. But if we closely examine the graph, we can see cities with bi-direction. For example cities 3 and 4 have directions to both sides.

- **Weighted Graphs**

Many graphs can have edges contain a weight associated to represent a real world implication such as cost, distance, quantity etc …



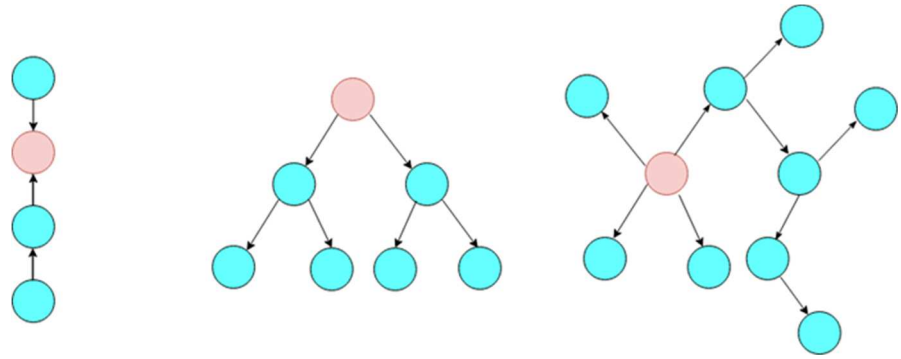
Weighted graphs could be either directed or undirected graph. The one we have in this example is an undirected weighted graph. The cost or distance from node green to orange and vice versa is 3. We could represent this relationship as a triplet like **(u, v, w)** which shows from where the edge is coming in, where it goes and cost or distance between the two nodes. Like our previous example, if you want to travel between two cities say city green and orange, we would have to pay off a cost of 3$ or in other terms, we would have to drive 3 miles. These metrics are self-defined and could be changed according to the situations. For a more elaborated example, consider you have to travel to city pink from green. If you look at the city graph, we can't find any direct roads or edges between the two cities. So what we can do is to travel via another city. The most promising routes would be starting from green to pink via orange and blue. If the weights

are costs between cities, we would have to spend 11$ to travel via blue to reach pink but if we take the other route via orange, we would only have to pay 10$ for the trip.

D. 12. 3. (*Special Graphs*):

- **Tree**

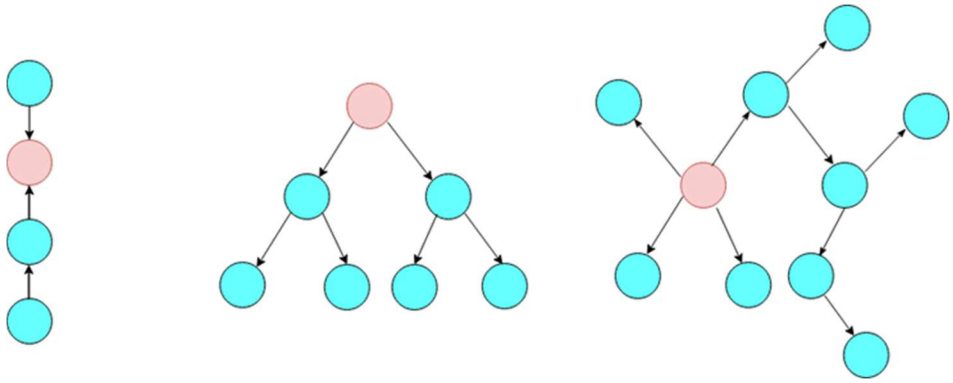
The most important type of special graphs is a tree. It's an undirected graph with no cycles. Equivalently, it has **N** nodes and **N — 1** edges.



All the graphs given above are trees even the left most one because it has no cycles in it.

- **Rooted Tree**

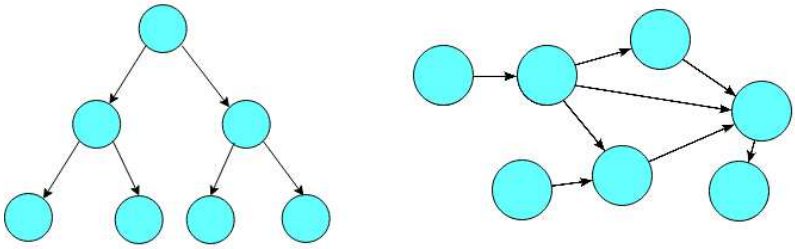
A rooted tree is a tree with a designated root node where all other nodes are either coming towards the root or going away from the root.



The node which is in red color is the root node. The left most tree is called an ***In-tree*** because all other nodes are coming towards the root node. The two other trees are ***Out-trees***, because all other nodes are going away from the root.

- **Directed Acyclic Graphs (DAGs)**

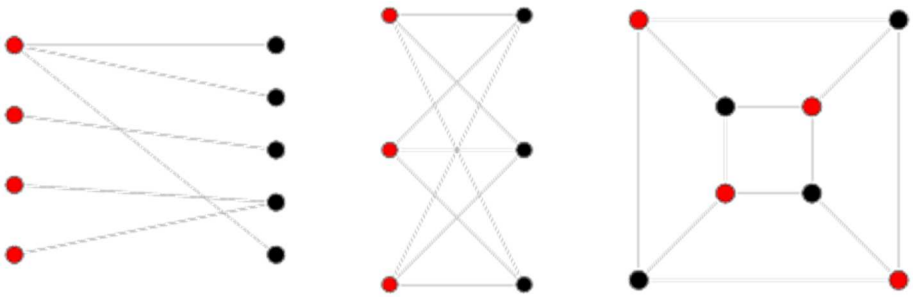
  DAGs are **directed graphs with no cycles**. These graphs play an important role in representing structures with dependencies such as schedulers and compilers.



  We can use this graph to show the topological order between something meaningful. For example, if we use DAGs in a process manager, we could say like sub-processes x and y should be completed before proceeding with z.

- **Bi-Partite Graphs**

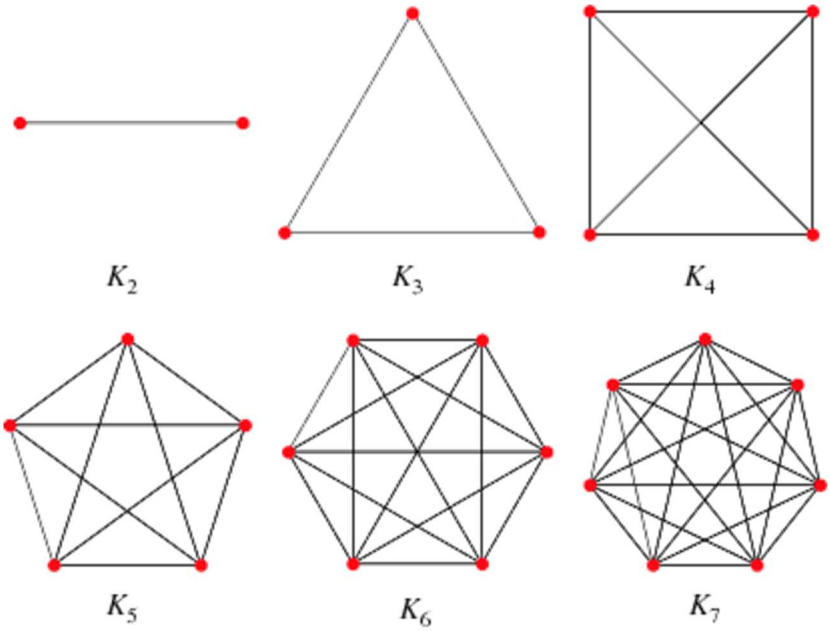
  A Bi-Partite Graph is a one whose vertices could be divided into two disjoint sets say U and V, where each edge in the graph connects the vertices between U and V. Other similar definition to a Bi-Partite Graph is that the graph would be two colourable.



  If we look at the graph, we could see like each graph is divisible into two disjoint sets (U, V) and each edge connects the nodes between U and V.

- **Complete Graph**

  We call a graph Complete iff, each pair of vertices has a unique edge connecting between them. A complete graph with n vertices is denoted as **Kn.**



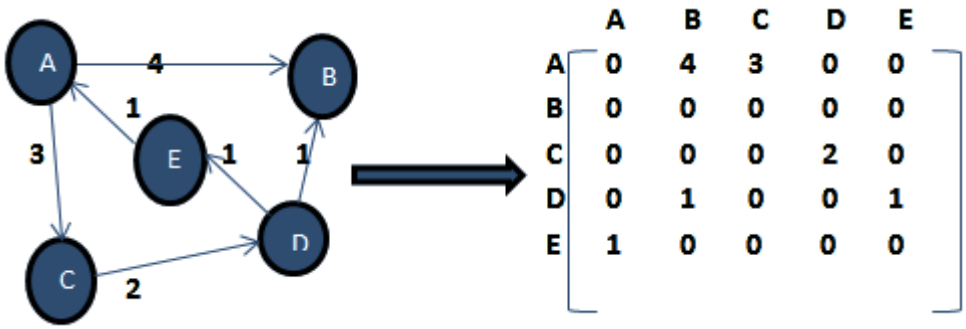
Complete graphs are considered to be the worst case graphs because of the number of edges to be traversed.

R. 12. 3. *(Representation of Graph)*

Here we discuss how we store a graph in the memory for further processes.

- **Adjacency Matrix**

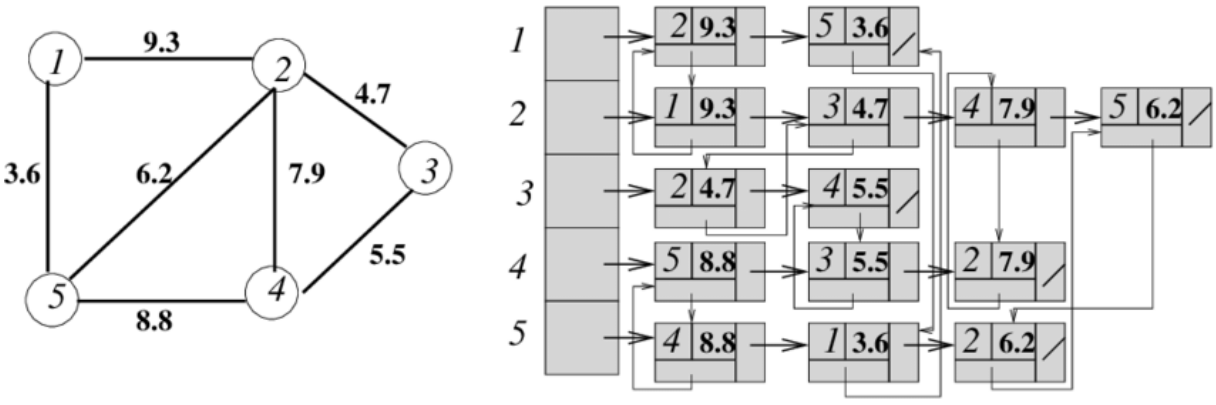
  The efficient way is to use a matrix of size **NxN** where N is the number of nodes. We call this matrix an *adjacency matrix.*



The above given is of a directed weighted graph and it's corresponding adjacency matrix **M**. The matrix is of size 5x5 as there are 5 nodes in total. The cost from node A to B is 4 and it's given in **M[A][B].** Similarly, the cost from one node to itself is zero, so all the diagonal elements would always be zeros. This is a space efficient method to store the graph information for the dense structures and the edge look up would always take a constant time. But, as the number of nodes increase, the space required to keep track of the edge information would also increase exponentially. If most of the edges don't have a proper information, we would endup making a sparse matrix.

- **Adjacency Lists**

  Another important structure we use to store the node and edge information is an adjacency list. This is a map from nodes to lists of edges.



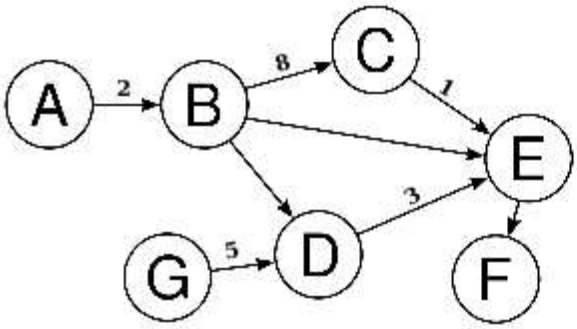
The given example is of an undirected weighted graph. If we look at the rightmost diagram, we could see couple of lists starting from each vertices. Node 1 has two outgoing edges namely 2 and 5, we represent this

information along with it's cost. Each element in the list has the target node and the corresponding cost or weight.

Adjacency List is an efficient mechanism to store the graph information if it is sparse i.e. it would take less memory as compared to the adjacency matrix in the same scenario. But still, this is a slight more complex representation rather than the adjacency matrix.

- **Edge Lists**

  AN edge list is a way to represent a graph simply as an unordered list of edges. Assume the notation for any triplet (u, v, w) means:
  " *the cost from u to v is w*".



[(A, B, 2), (B, C, 8), (C, E, 1), (D, E, 3), (G, D, 5)]

The edge list is given in the right hand side corresponding to the directed weighted graph on the left hand side. Each pair in the list shows edge information between two nodes and associated weight.